

FAKE JOB DETECTION USING MACHINE LEARNING



A PROJECT REPORT

Submitted by

M Vaibhav Kumar

UID: 20BCS4398

Ritik

UID: 20BCS4361

Gaurav Kumar Tiwary

UID: 20BCS3986

Abhimanyu

UID: 20BCS3975

*In partial fulfillment for the award of the degree
of*

**BACHELOR OF ENGINEERING
in
BE-CSE BIG DATA AND ANALYTICS**

Under the Guidance

Of

Ms. Rituparna Seal



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

APEX INSTITUTE TECHNOLOGY

CHANDIGARH UNIVERSITY, GHARUAN, MOHALI-140413,

PUNJAB

APRIL 2024

BONAFIDE CERTIFICATE

Certified that this project report “**FAKE JOB DETECTION USING MACHINE LEARNING**” is the bonafide work of “**FOUR**” candidates

M Vaibhav Kumar

UID: 20BCS4398

Ritik

UID: 20BCS4361

Gaurav Kumar Tiwary

UID: 20BCS3986

Abhimanyu

UID: 20BCS3975

who carried out the work under my supervision. Certified further that to the best of my knowledge the work reported herein does not form part of any other thesis or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE

Mr. Aman Kaushik

HEAD OF THE DEPARTMENT

SIGNATURE

Ms.Rituparna Seal

SUPERVISOR

CERTIFICATE FOR EVALUATION

College Name : Apex Institute of Technology

Branch : Big data and Analytics

Year / Semester: IV / VII

S.NO	Name of the Students Who have done the project	Title of the project	Name of the Supervisor with Designation
1.	M Vaibhav Kumar	FAKE JOB DETECTION USING MACHINE LEARNING	
2.	Ritik		
3.	Gaurav Kumar Tiwary		
4.	Abhimanyu		

The reports of the project work submitted by the above students in partial fulfillment for the award of Bachelor of Engineering degree in CSE Big Data and Analytics of Chandigarh University were evaluated and confirmed to be the reports of the work done by the above students and then evaluated.

INTERNAL EXAMINER

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

In the accomplishment of completion of our project on **FAKE JOB DETECTION USING MACHINE LEARNING**. We want to express our sincere gratitude to **Ms. Rituparna Seal**, our instructor and project supervisor at AIT-CSE Chandigarh University.

We appreciate all of your help, advice and suggestions, which we used throughout this project completion. In this regard, we shall always be grateful to you.

We are making sure that this project was completed by us without any plagiarism.

DECLARATION

We, Abhimanyu, M Vaibhav Kumar, Ritik, and Gaurav, students of ‘Bachelor of Engineering in Computer Science Engineer (Hons.) – Big Data and Analytics’, session: 2023-24, Department of Computer Science and Engineering, Apex Institute of Technology, Chandigarh University, Punjab, hereby declare that the work present in this Project Work entitled ‘Fake Job Detection using Machine Learning’ is the outcome of our own bona fide work and is correct to the best of our knowledge and this work has been undertaken taking care of Engineering Ethics. It contains no material previously published or written by another person nor material which has been accepted for the award of any other degree or diploma of the university or other institute of higher learning, except where due acknowledgement has been made in the text.

Abhimanyu 20BCS3975

M Vaibhav Kumar 20BCS4398

Ritik 20BCS4361

Gaurav Kumar Tiwary 20BCS3986

Date: November 2023

Place: Chandigarh University

ABSTRACT

The increasing prevalence of fake job advertisements in the digital job market has become a pressing concern, leading to financial loss and potential harm to job seekers. This project focuses on developing a robust fake job detection system using machine learning techniques. The objective is to accurately differentiate between genuine and fake job postings, aiding in the prevention of fraudulent activities.

The research begins with an extensive literature review, analyzing seven relevant papers in the field of fake job detection. Based on this review, the most used models are compared and then the project proceeds to propose a system architecture that incorporates Machine Learning techniques, such as TF-IDF for feature extraction, and the implementation of the passive aggressive classifier as the machine learning model. The dataset used for training and testing the system consists of a diverse collection of job postings. Through preprocessing and data cleaning techniques, the dataset is prepared for model training and evaluation. The performance of the system is assessed by comparing multiple machine learning models, including logistic regression, random forest, support vector machines (SVM), and the passive aggressive classifier. The experimental results demonstrate the effectiveness of the proposed system in accurately identifying fake job postings. Among the evaluated models, the passive aggressive classifier achieves the highest accuracy of 90.49%. This highlights its capability to discern between genuine and fake job advertisements with a high degree of accuracy.

The developed system is implemented in a user interface where users can input job descriptions and receive real-time results indicating whether a job posting is fake or genuine. The integration of HTML and CSS enhances the user experience, providing a visually appealing and intuitive interface. The project also explores the practical deployment of the system by utilizing Flask, a popular web framework, for building the backend. This ensures scalability, flexibility, and ease of deployment in real-world scenarios.

TABLE OF CONTENTS

Contents

BONAFIDE CERTIFICATE	2
CERTIFICATE FOR EVALUATION	3
ACKNOWLEDGEMENT	4
DECLARATION	5
ABSTRACT	6
1. INTRODUCTION	10
1.1 Problem Identification	10
1.2 Task Identification	12
1.3 Timeline	14
1.4 Organization of the report	15
CHAPTER 2	16
2.1 LITERATURE REVIEW	16
2.2 Literature Review Summary	19
CHAPTER 3	21
3. DESIGN FLOW/PROCESS	21
3.1 Software Used	14
3.1.1 PyCharm	14
3.1.2 Jupyter	14
3.1.3 Kaggle	14
3.1.4 Flask	14
3.2 Theoretical Background	14
3.2.1 Machine Learning	14
3.2.2 Logistic Regression	14
3.2.3 Support Vector Machines(SVM)	14
3.2.4 Random Forest	14
3.2.5 Passive-Aggressive Classifier	14
3.2.6 Flask	14
3.3 Proposed System	14
3.4 Diagrams	14

3.5 Description of the software used	14
3.5.1 The Front-End User Interface.....	14
3.5.2 Flask Server	14
3.5.3 Feature Extraction with TF-IDF	14
3.5.4 The Machine Learning Model	141
3.6 Code Snippets.....	50
3.6.1 HTML.....	51
3.6.2 CSS.....	14
3.6.3 API-Flask.....	14
3.6.4 Python.....	14
CHAPTER 4.....	64
4 RESULT ANALYSIS AND VALIDATION.....	64
4.1 Experimental Setup	14
4.1.1 Literature Review	14
4.1.2 Model Selection.....	14
4.1.3 Data Collection.....	14
4.1.4 Data Preprocessing	14
4.1.5 Feature Extraction	14
4.1.6 Model Training and Validation	14
4.1.7 Result Analysis.....	14
4.1.8 Experimental Validation.....	71
4.2 Model comparision and model selection.....	14
4.3 Results Snapshot.....	14
CHAPTER 5.....	77
5 CONCLUSION AND FUTURE WORK.....	77
REFERENCES	80

Table of figures/tables

Figure 1: Use Case Diagram.....	47
Figure 2: Class Diagram	47
Figure 3: Sequence Diagram.....	48
Figure 4: Schema Diagram.....	48
Figure 5.....	51
Figure 6.....	52
Figure 7.....	52
Figure 8.....	53
Figure 9.....	54
Figure 10.....	54
Figure 11.....	55
Figure 12.....	55
Figure 13.....	56
Figure 14.....	56
Figure 15.....	57
Figure 16.....	57
Figure 17.....	58
Figure 18.....	58
Figure 19.....	59
Figure 20.....	59
Figure 21.....	60
Figure 22.....	60
Figure 23.....	61
Figure 24.....	61
Figure 25.....	62
Figure 26.....	62
Figure 27.....	63
Figure 28.....	63
Figure 29: Basic Interface	73
Figure 30: Fraudulent Job.....	74
Figure 31: Real Job	75
Figure 32: Terminal	76

Table 1: Literature Review Summary	20
------------------------------------	----

CHAPTER 1

1. INTRODUCTION

1.1 Problem Identification

Job scams are a mounting problem in today's technology-driven world. With the augment of remote work and online job portals, almost all individuals are turning to the internet to find employment opportunities. Unfortunately, this has also led to an enhancement in job scams, where fraudsters craft fake job postings to lure unsuspecting victims into giving away personal information or paying a huge sum for non-existent jobs.

The dilemma with such scams is that they can be tricky to detect since they pretend to be from a valid source. Although various job portals previously warn job seekers regarding some common pointers of fake job postings and the masterminds behind them, the fraudsters are becoming gradually more refined in their tactics, making it convoluted for job seekers to make a distinction between genuine job postings and fake ones. This can lead to individuals wasting both time and money on deceitful job applications, as well as putting their personal information in jeopardy.

There are also evident cases of Cybercrimes related to the same scenario where job seekers are asked to fill in particularly sensitive information such as Pan card or Aadhar card details, bank details, and so on. As soon as the fraudsters acquire the extremely sensitive information they first request the job seekers for a certain amount which if denied, they tend to threaten the individuals claiming that if the respective amount is not paid they would misuse their information in unethical and illegal deeds.

Furthermore, to date, there are various machine learning models available which can be used to detect fake job postings. However, it is difficult to understand which model works best for the fake job detection purpose. Also, a system for job seekers is not developed keeping in mind users with a non-technical background, this means that there is no such system where any user can detect the legitimacy of a job posting with a user-friendly interface.

To address this crisis, our project aims to build a fake job detector using machine learning with a user interface where users can copy and paste a job description to know if it is real or fake. The main goal of this project is to build a tool that can facilitate individuals to protect themselves from job scams. By developing a machine learning model that can detect fake job postings, this project aims to provide job seekers with a powerful tool to protect themselves from fraud. The user interface developed for this project will make it easy for users to promptly verify whether a job posting is authentic or not, reducing the casualty of job scams.

1.2 Task Identification

Our project work centers around the development of a fake job detection system utilizing Machine Learning techniques. Our main objective was to review existing literature in the field and subsequently devise and implement our own model for enhanced accuracy and efficiency. To initiate our project, we conducted a comprehensive review of relevant papers, encompassing seven studies that explored diverse aspects of fake job detection with Machine Learning. This literature review allowed us to gain valuable insights into current methodologies and identify the most used Machine Learning models and techniques for fake job detection.

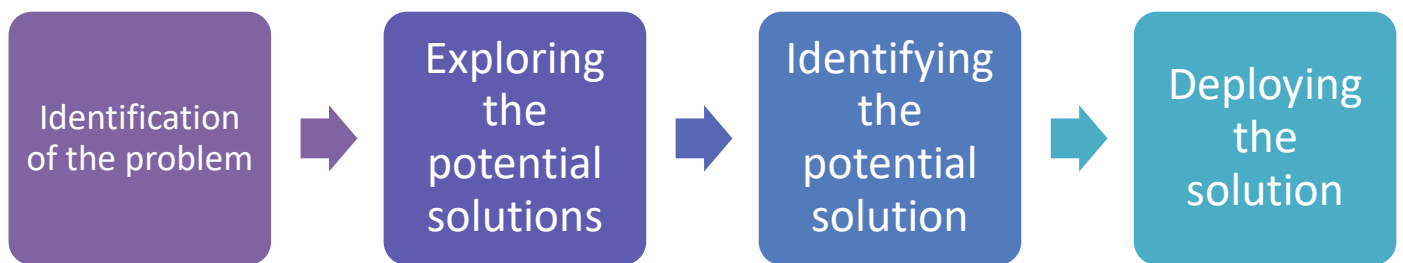
Building upon this foundation, we developed our own model for fake job detection, considering a range of Machine Learning algorithms and evaluating their performance. After careful analysis and comparison of the most used Machine Learning models and techniques like the Logistic Regression, Random Forest, SVM, and Passive Aggressive model, we determined that the passive aggressive model exhibited the highest accuracy and displayed great potential for effective classification. Consequently, we adopted this model as the cornerstone of our research, guiding the development of our fake job detection system.

Additionally, we presented our review paper in the National Conference of Research in Emerging Areas (NACORE), 2023 on April 26th. The conference recognized the value and contribution of our work, and as a result, our review paper has been accepted for publication in the esteemed International Journal of Engineering Research and Applications.

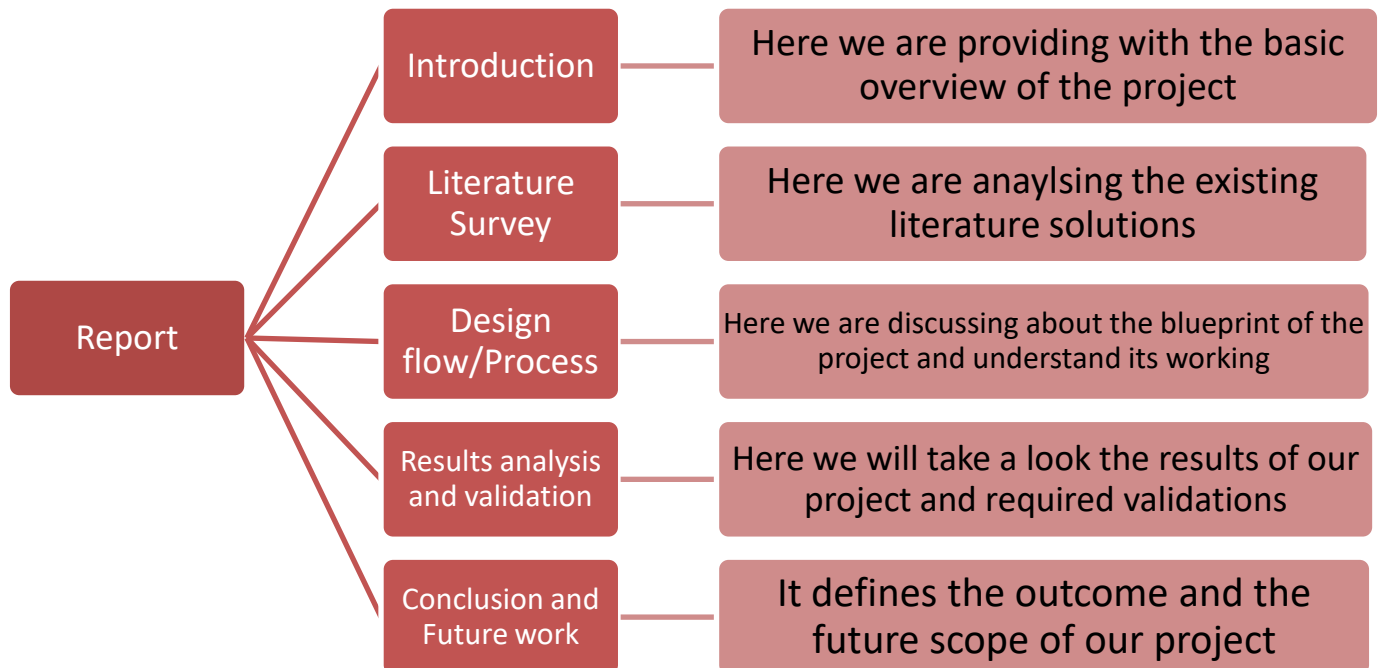
Moreover, as part of our effort to enhance usability and accessibility, we implemented a user interface that enables users to input job descriptions and receive real-time results indicating whether a job is fake or genuine. This interface serves as a practical solution to aid job seekers in efficiently identifying fraudulent job postings. By integrating insights from the literature review, developing our own model, and creating a user-friendly interface, our research aims to strengthen the reliability and integrity of online job platforms, benefitting both job seekers and employers.

In conclusion, our project work involved an in-depth review of existing research papers on fake job detection using Machine Learning. Subsequently, we formulated our own model, with the passive aggressive algorithm at its core, to establish an accurate and accessible system. We are proud to have presented our research at a prestigious National Conference and are excited to have our review paper accepted for publication in the International Journal of Engineering Research and Applications. Through these contributions, our research seeks to tackle the challenges posed by fake job postings and advance the realm of online job marketplaces.

1.3 Timeline



1.4 Organization of the report



CHAPTER 2

2.1 LITERATURE REVIEW

Baraneetharan (2022) used a dataset of 45,465 job advertisements, in the paper proposed a machine learning-based approach to detect fake job advertisements using the Feature engineering technique. In the research, the features that were used to achieve the goal were job titles, company names, and job descriptions. The techniques that were used were decision trees, random forest, and logistic regression algorithms to train and test the model. While all the techniques that were used gave a satisfactory result it was found that the best performance of all three was obtained with the decision tree algorithm. The decision tree algorithm gave an accuracy of 96.2% which even if is not the highest among the other researchers, is quite satisfactory and gave pretty good results. [1]

Ullah (2023) proposed a smart and secure framework for detecting and averting online recruitment fraud in the paper. In the research paper ensemble Machine learning techniques were used. Feature extraction techniques were also used in this paper, and the author used various job features such as job titles, job descriptions, and company names to train the model. To improve the performance of the model, the author also made use of techniques like bagging and boosting. The proposed model achieved an accuracy of 92.5%. Considering the less amount of datasets that were taken which is only 10,000, the results were agreeable. [2]

Naude, Adebayo, and Nanda (2022), in their research, proposed a Machine learning approach to detect fraudulent job types. In their research work, the authors tried to use and implement unsupervised learning techniques such as k-means clustering to cluster job advertisements and detect fraudulent jobs. It is the method by which the unlabeled

dataset is grouped into different clusters based on some rules. The model that was

proposed by the authors had an accuracy of 89%, which is the lowest compared to the other researchers. However, the technique that has been proposed is highly efficient and if used on a higher number of datasets can potentially achieve a higher accuracy rate. [3]

Vo and Sharma (2021) proposed a method to deal with the class imbalance problem in the detection of fake job descriptions particularly. They proposed a technique called "SMOTE" (Synthetic Minority Over-sampling Technique) to overcome this problem. In this technique, the authors used a combination of oversampling and undersampling to balance the dataset and improve the performance of the models.

Feature extraction techniques were also used in the proposed model by the authors to train the model. The proposed approach achieved an overall accuracy of 95% in detecting fake job descriptions, using the Random Forest algorithm. The precision, recall, and F1-score were also high for both the minority class (fake job descriptions) and the majority class (genuine job descriptions). [4]

Amaar et al. (2022) proposed a machine learning and natural language processing (NLP) approach to detect fake job postings. The authors performed feature engineering, data cleaning, and data preprocessing before applying machine learning algorithms. The authors used two datasets and both datasets were preprocessed and transformed into numerical features before being used to train and evaluate the machine learning models. The authors used several machine learning algorithms, including Naive Bayes, Decision Tree, Random Forest, and Gradient Boosting, to evaluate the performance of their proposed approaches. On the first dataset, which contains 17,880 job postings, out of which 866 were labeled as fake, the proposed approach achieved an accuracy of 96.1% using the Random Forest classifier. On the second dataset, which contains 11,000 job postings with 7,828 labeled as genuine and 3,172 labeled as fake, the proposed approach

achieved an accuracy of 97.8% using the Gradient Boosting classifier. [5]

In the study by Kumar (2020), the author used machine learning algorithms to classify fake and real job advertisements. The author applied different machine learning algorithms, such as Vector Classifier, decision tree, and random forest, and evaluated the performance of each algorithm. The Support Vector Classifier gave an accuracy score of 97.78% for 5354 test observations. It is also noteworthy that it was able to correctly predict the class labels for 5245 job postings. [6]

The proposed approach in the paper by Dutta and Bandyopadhyay (2020), used several machine learning algorithms, including Naive Bayes, Decision Tree, Random Forest, Support Vector Machine (SVM), and K-Nearest Neighbor (KNN). The paper reported the accuracy of each algorithm on the testing set, among which experimental results indicate that the Random Forest classifier outperforms its peer classification tool. The proposed approach achieved an accuracy of 98.27% which is much higher than the existing methods. [7]

2.2 Literature Review Summary

Sl. No.	Paper Name	Authors	Publishing date	Feature Extraction	Model	Accuracy
	Detection of Fake Job Advertisements using Machine Learning algorithms	E. Baraneetharan	14.10.2022	Title, Department, Corporate Profile, Summary, Condition, Rewards, Work from home, firm logo, Employment type, necessary education, necessary experience, industry, function	Three models KNN, SVM, XGboost were used, but XGboost was found out to be most efficient	98.53%
	A smart secured framework for detecting and averting online recruitment fraud using ensemble machine learning techniques	Zahid Ullah, Mona Jamjoom	08-02-2023	Job_id, title, location, department, salary range, company profile, description, requirements, benefits, company logo, has_questions, employment type, required_experience, required_education, industry, function	SVM, DT, Logistic Regression, and k-NN	98.374%
	Dealing with the class imbalance problem in the detection of fake job description	Anh Vo, Rohit Sharma	03-2021	Bag of words, TF-IDF	Logistic Regression	0.93 False positive rate

	A machine learning approach to detecting fraudulent job types	Marcel Naude, Kolawole John Adebayo, and Rohan Nanda	25-05-2022	Bag of words, Handcrafted binary rule-set based features model	Random forest classifier	91%
	Detection of Fake Job Postings by Utilizing Machine Learning and Natural Language Processing Approaches	Aashir Amaar, Wajdi Aljedaani, Furqan Rustam, Saleem Ullah, Vaibhav Rupapara & Stephanie Ludi	04-01-2022	Bag of words, TF-IDF	Adaptive synthetic sampling approach (ADASYN)	99.9%
	Classifying Fake and Real Job Advertisements using Machine Learning	Dr. Vaibhav Kumar	26-06-2020	Tf idf Vectorizer, Count Vectorizer	Support Vector Classifier	97.78%
7.	Fake Job Recruitment Detection Using Machine Learning Approach	Shawni Dutta and Prof.Samir Kumar Bandyopadhyay	04-2020	Job_id, title, location, department, salary_range, company profile, requirements, description, benefits, telecommuting, has_company_logo, has_questions, employment type, required experience, required education, industry, function, fraudulent	Random Forest classifier	98.27%

Table 1: Literature Review Summary

CHAPTER 3

3. DESIGN FLOW/PROCESS

3.1 Software Used

3.3.1 PyCharm

Anaconda is an open-source distribution of the Python and R programming languages for data science and machine learning tasks. It includes a package manager, environment management, and numerous scientific computing libraries, such as NumPy, Pandas, Matplotlib, and Scikit-learn.

Anaconda provides an integrated development environment (IDE) called Anaconda Navigator, which simplifies the installation and management of libraries and packages. Additionally, Anaconda allows users to create virtual environments with specific library versions, which helps avoid conflicts between different projects.

Overall, Anaconda is a powerful tool for data scientists, researchers, and developers who need a comprehensive and streamlined environment for data analysis and machine learning projects.

PyCharm Features

- Anaconda offers a package management, environment manager, and data science tools.
- Pandas, NumPy, Matplotlib, Scikit-learn, TensorFlow, and Jupyter Notebook are among its 1,500 data science tools.
- Anaconda lets users establish and maintain several environments with distinct package dependencies and versions, making
- it easier to switch projects and avoid package conflicts.
- Anaconda Navigator, its graphical interface, makes managing packages, environments, and Jupyter notebooks easy.
- Anaconda offers enterprise-level collaboration, version control, and deployment tools for massive data science projects.

Key Features of PyCharm:

- **Code Editor:** PyCharm provides a powerful code editor with features like syntax highlighting, code completion, code analysis, and code navigation. It supports not only Python but also other languages and technologies.
- **Code Assistance:** PyCharm offers intelligent code completion, helping you write code faster and with fewer errors. It suggests code fixes, quick fixes, and code inspections to help you write clean and maintainable code.
- **Integrated Debugger:** The built-in debugger allows you to set breakpoints, inspect variables, and step through your code, making it easier to find and fix bugs.

- **VCS (Version Control System) Integration:** PyCharm seamlessly integrates with popular version control systems like Git, Mercurial, and Subversion. You can commit, pull, and push changes right from the IDE.
- **Virtual Environment Support:** It supports creating and managing virtual environments for your Python projects, which is essential for isolating project dependencies and avoiding conflicts.
- **Database Tools:** PyCharm includes database tools that allow you to connect to various databases, run SQL queries, and view and edit data.
- **Web Development Support:** PyCharm provides support for web development technologies like HTML, CSS, and JavaScript. You can also develop web applications with frameworks like Django and Flask.
- **Scientific Tools:** PyCharm is a great choice for data science and scientific computing. It supports Jupyter notebooks, data visualization libraries like Matplotlib, and scientific libraries like NumPy and SciPy.
- **Testing Frameworks:** PyCharm supports various testing frameworks such as unittest, pytest, and nose. You can run tests and view the results within the IDE.
- **Code Templates and Snippets:** It offers code templates and code snippets to help you write repetitive code more quickly.
- **Customizable UI:** PyCharm allows you to customize the user interface, themes, and keybindings to suit your preferences.

- **Plug-in Ecosystem:** You can extend PyCharm's functionality through a rich ecosystem of plugins and extensions, making it suitable for a wide range of development tasks.

Usage of PyCharm:

- **Creating a New Project:** When you launch PyCharm, you can create a new Python project, specify its location, and choose the Python interpreter (including virtual environments).
- **Editing Code:** You can write, edit, and organize your Python code within the code editor. PyCharm provides features like autocompletion, code analysis, and quick fixes.
- **Running and Debugging Code:** You can run your Python code within PyCharm, set breakpoints, and use the integrated debugger to step through code and inspect variables.
- **Version Control:** PyCharm integrates with version control systems, allowing you to manage your code repository, commit changes, and collaborate with others.
- **Database Management:** If you are working with databases, you can connect to them from within PyCharm, run SQL queries, and manage your database.
- **Web Development:** PyCharm provides features for web development, including support for HTML, CSS, JavaScript, and popular web frameworks.
- **Data Science and Scientific Computing:** PyCharm is used for data science projects, including Jupyter notebook support and integration with scientific libraries.
- **Testing:** You can write and run tests using various testing frameworks, and PyCharm provides test results and coverage reports.
- **Customization:** You can customize the IDE's appearance, keybindings, and functionality by installing plugins and themes.

- Deployment: PyCharm allows you to deploy your applications and web services to various platforms and servers.

In summary, PyCharm is a versatile and powerful IDE for Python development. Its rich set of features and integrations make it a popular choice for both beginners and experienced Python developers. It helps streamline the development process and ensures a productive coding experience.

3.3.2 Jupyter

Jupyter is a project and community whose goal is to "develop open-source software, open standards, and services for interactive computing across dozens of programming languages". It was spun off from IPython in 2014 by Fernando Perez and Brian Granger. Project Jupyter's name is a reference to the three core programming languages supported by Jupyter, which are Julia, Python and R, and also a homage to Galileo's notebooks recording the discovery of the moons of Jupiter. Project Jupyter has developed and supported the interactive computing products Jupyter Notebook, JupyterHub, and JupyterLab. Jupyter is fiscally sponsored by NumFOCUS. JupyterLab is the latest web-based interactive development environment for notebooks, code, and data. Its flexible interface allows users to configure and arrange workflows in data science, scientific computing, computational journalism, and machine learning. A modular design invites extensions to expand and enrich functionality. The Jupyter Notebook is the original web application for creating and sharing computational documents. It offers a simple, streamlined, document-centric experience.

Jupyter is an open-source interactive computing environment used for creating and sharing documents that contain live code, equations, visualizations, and narrative text. The name "Jupyter" is a combination of three core programming languages it supports: Julia, Python, and R. It was formerly known as IPython (Interactive Python) Notebook before expanding its language support.

Key Components of Jupyter:

- **Jupyter Notebook:** The Jupyter Notebook is the primary interface for working with Jupyter. It's a web-based application that allows you to create and interact with documents known as "notebooks."
- **Kernels:** Kernels are the computational engines behind Jupyter. Each notebook is associated with a specific kernel that executes the code within the notebook. Jupyter supports a wide range of programming languages, and different kernels allow you to use different languages in the same Jupyter environment.
- **Cells:** Notebooks are composed of cells, which can contain code, text, or mathematical equations. There are two main types of cells:
 - **Code Cells:** These cells are used for writing and executing code. You can write code in a code cell, and when you run the cell, the output is displayed immediately below it.
 - **Markdown Cells:** These cells are used for adding narrative text, documentation, and explanations using Markdown markup. Markdown cells support formatted text, headers, lists, links, and more.
- **Interactive Execution:** Jupyter provides an interactive computing environment, which means you can execute code cells one by one and see the results in real-time. This is particularly useful for data analysis, exploration, and visualization.
- **Rich Output:** Jupyter notebooks support the display of rich output, including plots, charts, images, interactive widgets, and more. This makes it a valuable tool for data visualization and scientific computing.
- **Magic Commands:** Jupyter supports special commands called "magic commands" that provide enhanced functionality for code cells. For example, `%matplotlib inline` is a magic command that enables inline plotting in notebooks.
- **Extensions:** Jupyter has a vibrant ecosystem of extensions that enhance its functionality. These extensions include Jupyter widgets for interactive UI elements, extensions for theming and customizing the notebook interface, and more.

Usage of Jupyter

- **Data Analysis and Visualization:** Jupyter is widely used in data science and data analysis for tasks like data cleaning, exploration, and visualization. Libraries like Pandas, Matplotlib, Seaborn, and Plotly are commonly used within Jupyter notebooks.
- **Machine Learning and Deep Learning:** Jupyter is a popular environment for machine learning and deep learning projects. Users can build and train machine learning models using libraries like scikit-learn, TensorFlow, and PyTorch.
- **Education and Teaching:** Jupyter is used in educational settings to create interactive lessons, tutorials, and assignments. It's an excellent tool for teaching programming, data science, and other technical subjects.
- **Research and Scientific Computing:** Scientists and researchers use Jupyter for conducting experiments, running simulations, and documenting their work. It's particularly well-suited for research projects in fields like physics, biology, and engineering.
- **Report Generation:** Jupyter notebooks can be converted to various formats, including HTML, PDF, and LaTeX, making it a convenient tool for generating reports and research papers.
- **Sharing and Collaboration:** Jupyter notebooks can be easily shared with others. They can be published online, making it straightforward to collaborate and share findings with colleagues and the broader community.
- **Automating Tasks:** Jupyter notebooks can be used for automating repetitive tasks, data processing, and data extraction, and are often used in business and data engineering applications.

In summary, Jupyter is a versatile platform for interactive and exploratory computing. Its support for multiple programming languages, interactive execution, and rich output capabilities make it a powerful tool for data analysis, scientific research, education, and more. It has gained widespread popularity in the fields of data science, machine learning, and research due to its flexibility and ease of use.

3.3.3 Kaggle

Kaggle, a subsidiary of Google LLC, is an online community of data scientists and machine learning practitioners. Kaggle allows users to find and publish data sets, explore and build models in a web-based data-science environment, work with other data scientists and machine learning engineers, and enter competitions to solve data science challenges.

Kaggle got its start in 2010 by offering machine learning competitions and now also offers a public data platform, a cloud-based workbench for data science, and Artificial Intelligence education. Its key personnel were Anthony Goldbloom and Jeremy Howard. Nicholas Gruen was founding chair succeeded by Max Levchin. Equity was raised in 2011 valuing the company at \$25 million. On 8 March 2017, Google announced that they were acquiring Kaggle. Kaggle has run hundreds of machine learning competitions since the company was founded. Competitions have ranged from improving gesture recognition for Microsoft Kinect to making a football AI for Manchester City to improving the search for the Higgs boson at CERN.

Kaggle, established in 2010, has evolved into a global powerhouse in the data science and machine learning arena, and it's at the forefront of fostering a vibrant data community. Its array of components provides a comprehensive ecosystem for data enthusiasts. At the heart of Kaggle are its datasets, an extensive collection spanning diverse domains such as finance, healthcare, and sports, which users can access, download, and employ for analysis or machine learning projects. The real allure of Kaggle, however, lies in its renowned data science competitions. These challenges invite participants to develop predictive models or data-driven solutions, often in response to real-

world problems posed by organizations. The competitions offer not just intellectual satisfaction but substantial cash prizes and opportunities like internships or job placements, making Kaggle a bustling marketplace for showcasing one's skills and innovations.

Kaggle Kernels and Notebooks are invaluable for data scientists and analysts. Kernels provide a cloud-based, interactive Jupyter Notebook environment that enables users to write and execute code directly connected to datasets. Users can run their analyses and models in the cloud without having to set up their own computing environment. Additionally, Kaggle Notebooks allow data professionals to seamlessly combine code, narrative text, and visualizations to create detailed project reports. This documentation and storytelling aspect is particularly useful for sharing insights and engaging with the Kaggle community.

Kaggle is not just about data and code; it's about fostering a sense of community and knowledge-sharing. The discussion forums are bustling hubs of knowledge exchange, where users can ask questions, seek help, and engage in meaningful discussions related to data science and machine learning. With a supportive and diverse user base, these forums are a fantastic resource for both novices and experts in the field.

For those looking to learn or upskill, Kaggle Learn offers a set of interactive courses and tutorials that cater to various levels of expertise. These resources are designed to help users master the fundamentals of data science and machine learning, with hands-on practice and guided instruction. Additionally, Kaggle Jobs connects data professionals with job opportunities in the field, ensuring that the platform is not just about learning and competing but also a gateway to promising career paths.

In summary, Kaggle's rich ecosystem, coupled with a dynamic and supportive community, makes it a pivotal platform for anyone involved in data science, machine learning, and data analysis. Whether you are seeking to sharpen your skills, collaborate with peers, solve real-world challenges, or find career opportunities, Kaggle provides the ideal environment for personal and professional growth in the realm of data.

3.3.4 Flask

Flask is a lightweight and versatile micro-framework written in Python. It is designed to make getting started with web development quick and easy, with the flexibility to scale up to complex applications. Flask is known for its simplicity and minimalism, providing the essential tools to build web applications without imposing any particular way of doing things. With its clear and concise syntax, Flask has become one of the most popular frameworks for Python web development.

Key Components of Flask

- **Routing:** Flask uses a decorator-based approach to define routes and map them to specific functions. This enables developers to create clean and organized URL structures, making the application more maintainable and easier to understand.
- **Templates:** Flask integrates the Jinja2 templating engine, allowing developers to render dynamic HTML content with ease. This facilitates the creation of dynamic web pages by inserting variables, loops, and control structures directly into the HTML templates.
- **HTTP Request/Response Handling:** Flask provides a request object to access incoming request data and a response object to build and send HTTP responses. This simplifies the handling of HTTP methods, headers, and cookies, making it easier to process and respond to client requests.
- **Extensions:** Flask has a rich ecosystem of extensions to add additional functionalities, such as form validation, database integration, and authentication. These

extensions can be easily integrated into Flask applications to enhance their capabilities and reduce development time.

- **Development Server:** Flask includes a built-in development server, making it easy to test and debug applications during development. The development server automatically reloads the application when changes are made, providing a seamless and efficient development experience.

Usage of Flask

Flask is widely used for developing a variety of web applications, ranging from simple websites and APIs to more complex web services and applications. Its simplicity and flexibility make it a popular choice among developers for various purposes:

- **Prototyping:** Developers often use Flask for quickly building and testing new web application ideas. Its minimalistic design and easy-to-use syntax allow developers to focus on the core functionality of the application without getting bogged down by unnecessary complexities.
- **Web Services and APIs:** Flask is ideal for developing lightweight and scalable web services and APIs with minimal overhead. Its simple and straightforward structure makes it easy to create RESTful APIs and handle HTTP requests and responses efficiently.

- **Full-Featured Web Applications:** With the integration of various extensions and libraries, Flask can be used to build complex web applications. Developers can easily add functionalities like database management, user authentication, and more, making it possible to develop robust and feature-rich web applications.

Flask's ease of use, clear and concise syntax, and rich ecosystem of extensions make it an excellent choice for both beginners and experienced developers looking for a flexible and efficient framework for web development. Its versatility and scalability allow it to be used for a wide range of web development projects, from small personal projects to large-scale commercial applications.

3.2 Theoretical Background

The theoretical foundation of the proposed fake job detection system incorporates essential principles and methodologies from machine learning, natural language processing, and web development. This segment offers a glimpse into the theoretical underpinnings supporting the design and execution of the system. The theoretical framework of the envisioned fake job detection system integrates fundamental principles and techniques from machine learning, natural language processing, and web development. Here, we present an outline of the theoretical groundwork that forms the basis for the system's design and deployment. The theoretical underpinning of the proposed fake job detection system integrates crucial concepts and approaches from machine learning, natural language processing, and web development. In this section, we offer an overview of the theoretical foundations that inform the design and implementation of the system.

3.2.1. Machine Learning:

Machine learning algorithms serve as the backbone of the fake job detection system, playing a pivotal role in its functionality. Among these algorithms, supervised learning algorithms stand out for their significance, empowering the system to glean insights from labeled data and extrapolate predictions onto unseen instances. Notably, logistic regression, support vector machines (SVM), random forests, and passive-aggressive classifiers are frequently employed in binary classification tasks such as fake job detection. By analyzing the provided data, these algorithms discern patterns and extrapolate them to ensure precise predictions on newly posted job listings.

At the heart of the fake job detection system lies a sophisticated array of machine learning algorithms, each contributing to its robust functionality. Of particular importance are supervised learning algorithms, which enable the system to leverage labeled data for learning purposes and extend its predictive capabilities to unseen instances. Logistic regression, support vector machines (SVM), random forests, and passive-aggressive classifiers emerge as key players in the realm of binary classification tasks, such as those encountered in fake job detection. Through meticulous analysis of the available data, these algorithms adeptly identify patterns and generalize them to ensure accurate predictions on newly surfaced job postings.

The fake job detection system owes much of its efficacy to the intricate interplay of various machine learning algorithms embedded within its framework. Among these algorithms, supervised learning algorithms take center stage, furnishing the system with the ability to harness labeled data for learning and prediction purposes. Logistic regression, support vector machines (SVM), random forests, and passive-aggressive classifiers are indispensable in the landscape of binary classification tasks, a domain where fake job detection resides. Through a meticulous examination of the provided data, these algorithms discern underlying patterns and extrapolate them to ensure precise predictions on freshly posted job listings.

3.2.2. Logistic Regression:

Logistic regression stands as a cornerstone in the realm of machine learning, particularly renowned for its efficacy in handling binary classification tasks. Its primary function entails modeling the intricate relationship between input features and the likelihood of

an instance being categorized into a specific class, such as distinguishing between fake and genuine job postings. Leveraging a logistic function, logistic regression adeptly maps the linear combination of input features to the probability of class membership. Through meticulous training facilitated by optimization techniques like gradient descent, the model iteratively refines its parameters to achieve optimal estimates that maximize the likelihood of the observed data.

When it comes to binary classification tasks, logistic regression emerges as a quintessential machine learning algorithm, revered for its versatility and performance. At its core, logistic regression meticulously models the intricate interplay between input features and the probability of an instance being assigned to a specific class, such as discerning between fake and genuine job postings. Employing a logistic function, this algorithm adeptly transforms the linear combination of input features into a probabilistic framework, facilitating informed decision-making. Through iterative training techniques, notably gradient descent, logistic regression fine-tunes its parameters to attain optimal estimates, thereby maximizing the likelihood of the observed data.

In the domain of binary classification tasks, logistic regression reigns supreme as a tried-and-tested machine learning algorithm, renowned for its robustness and versatility. Its primary objective revolves around modeling the nuanced relationship between input features and the probability of an instance being categorized into a specific class, such as differentiating between fake and genuine job postings. By leveraging a logistic function, logistic regression effectively transforms the linear combination of input features into a probabilistic framework, enabling nuanced decision-making. Through iterative optimization procedures, notably gradient descent, logistic regression iteratively adjusts

its parameters to achieve optimal estimates, thereby enhancing the likelihood of accurately capturing the observed data.

3.2.3. Support Vector Machines (SVM):

Support Vector Machines (SVMs) represent a formidable arsenal within the realm of supervised learning, demonstrating prowess in both linear and nonlinear classification endeavors. At their core, SVMs embark on a quest to identify an optimal hyperplane within a high-dimensional feature space, strategically positioned to maximize the margin between the two classes under consideration. Through a judicious mapping of input features into this augmented space, SVMs facilitate the discernment of intricate decision boundaries, crucial for tackling complex classification scenarios. Guided by optimization techniques, the model undergoes rigorous training to ascertain the hyperplane that best demarcates fake from genuine job postings, ensuring robust performance in real-world applications.

Support Vector Machines (SVMs) stand as stalwarts in the realm of supervised learning, revered for their adeptness in tackling both linear and nonlinear classification challenges with finesse. Operating within a high-dimensional feature space, SVMs embark on a quest to unearth an optimal hyperplane that aptly segregates the two classes under consideration, maximizing the margin between them. Through a deft mapping of input features into this augmented space, SVMs empower the identification of intricate

decision boundaries essential for navigating complex classification landscapes. Guided by optimization techniques, the model undergoes meticulous training to pinpoint the hyperplane that best delineates fake from genuine job postings, thereby ensuring robust performance and reliability in practical scenarios.

Within the domain of supervised learning, Support Vector Machines (SVMs) reign supreme, renowned for their proficiency in addressing both linear and nonlinear classification tasks with finesse. Central to their operation is the quest for an optimal hyperplane within a high-dimensional feature space, strategically positioned to maximize the margin between distinct classes. Through a sophisticated mapping of input features into this expanded space, SVMs facilitate the identification of intricate decision boundaries crucial for tackling complex classification scenarios. Driven by optimization techniques, the model undergoes rigorous training to ascertain the hyperplane that best distinguishes fake from genuine job postings, thereby ensuring robust performance and efficacy in real-world applications.

3.2.4. Random Forest:

Random Forest emerges as a formidable ensemble learning algorithm renowned for its prowess in harnessing the collective intelligence of multiple decision trees to deliver accurate predictions. In this sophisticated approach, each decision tree undergoes training on a random subset of the data, with their individual predictions subsequently aggregated to yield the final classification outcome. By leveraging this ensemble strategy, Random Forest effectively mitigates the risk of overfitting while enhancing prediction accuracy through the reduction of variance and bias inherent in individual

decision trees. This robust algorithm demonstrates remarkable versatility, capable of seamlessly navigating high-dimensional feature spaces and adeptly capturing intricate interactions between features, thereby cementing its status as a cornerstone in the domain of machine learning.

At the forefront of ensemble learning algorithms, Random Forest stands tall, heralded for its ability to harness the collective wisdom of numerous decision trees to furnish accurate predictions. Through a sophisticated orchestration, each decision tree within the Random Forest framework undergoes training on a random subset of the data, with their diverse predictions amalgamated to produce the final classification verdict. This ensemble approach not only serves as a bulwark against overfitting but also elevates prediction accuracy by curtailing the variance and bias inherent in individual decision trees. Exhibiting robustness and adaptability, Random Forest thrives in traversing high-dimensional feature spaces, adeptly capturing intricate feature interactions, and solidifying its stature as a cornerstone in the machine learning landscape.

In the realm of ensemble learning, Random Forest emerges as a stalwart, celebrated for its adeptness in harnessing the collective intelligence of multiple decision trees to deliver precise predictions. Underpinning its efficacy is a nuanced approach wherein each decision tree undergoes training on a random subset of the data, their diverse predictions amalgamated to yield the final classification outcome. By embracing this ensemble strategy, Random Forest effectively mitigates overfitting while bolstering prediction accuracy through the attenuation of variance and bias inherent in individual decision trees. Endowed with resilience and adaptability, Random Forest navigates high-dimensional feature spaces with ease, adeptly capturing complex feature interactions, and cementing its position as a linchpin in the realm of machine learning.

3.2.5. Passive-Aggressive Classifier:

The Passive-Aggressive classifier stands out as a distinctive variant of online learning algorithms tailored for tasks characterized by streaming data or constrained computational resources. Its utility shines in contexts where the classification model necessitates continuous updates to accommodate the influx of new job postings. Operating on an incremental basis, the classifier systematically refines the model with each new observation, dynamically adjusting its parameters to minimize the loss function. This inherent flexibility renders the Passive-Aggressive classifier well-suited for real-time fake job detection, empowering the system to seamlessly adapt to evolving trends and patterns as they unfold.

Within the realm of machine learning, the Passive-Aggressive classifier emerges as a notable iteration of online learning algorithms, ideally suited for scenarios featuring

streaming data or limited computational resources. Its efficacy becomes particularly pronounced in contexts where the classification model demands incessant updates to accommodate the continuous influx of new job postings. Leveraging an incremental learning approach, the classifier iteratively enhances the model's performance by adapting its parameters to minimize the loss function with each new observation. This inherent adaptability positions the Passive-Aggressive classifier as a formidable tool for real-time fake job detection, allowing the system to swiftly respond to shifting trends and emerging patterns.

In the realm of machine learning, the Passive-Aggressive classifier emerges as a versatile solution tailored for tasks characterized by streaming data or constrained computational resources.

Its efficacy is most pronounced in scenarios where the classification model necessitates frequent updates to accommodate the dynamic nature of new job postings. Employing an incremental learning paradigm, the classifier adeptly adjusts its parameters with each new observation, systematically minimizing the loss function to refine the model. This inherent adaptability renders the Passive-Aggressive classifier an invaluable asset for real-time fake job detection, enabling the system to adeptly navigate evolving trends and patterns with agility and precision.

3.2.6. Flask:

Flask stands as one of the most widely adopted Python web frameworks, revered for its versatility in crafting robust web applications. Armed with a rich array of tools and

libraries, Flask empowers developers to construct intuitive user interfaces and proficiently manage HTTP requests and responses. Within the proposed system, Flask emerges as a pivotal component, offering a streamlined avenue for creating a user-friendly interface. Through Flask, users can effortlessly input job descriptions and promptly receive real-time feedback on the authenticity of job postings. By seamlessly handling routing, request processing, and dynamic content rendering, Flask expedites the development process and facilitates the seamless integration of the fake job detection model into a web-based application environment.

In the realm of web development, Flask stands out as a cornerstone Python framework renowned for its simplicity and flexibility. Equipped with a comprehensive suite of tools and libraries, Flask empowers developers to craft immersive web applications with ease. Within the context of the proposed system, Flask assumes a central role in facilitating the creation of a user-friendly interface. Through Flask, users can conveniently input job descriptions and promptly receive real-time feedback on the legitimacy of job postings.

By handling crucial tasks such as routing, request handling, and dynamic content rendering, Flask streamlines the development process and facilitates the seamless integration of the fake job detection model into a web-based application.

Renowned for its minimalist design and unparalleled flexibility, Flask stands as a premier Python web framework, empowering developers to create elegant and efficient

web applications. Armed with an extensive suite of tools and libraries, Flask provides developers with the necessary resources to build intuitive user interfaces and manage HTTP requests and responses seamlessly. In the envisioned system, Flask assumes a pivotal role, serving as the backbone for developing a user-friendly interface. Through Flask, users can effortlessly input job descriptions and receive instantaneous feedback on the authenticity of job postings in real-time. By handling essential tasks such as routing, request processing, and dynamic content rendering, Flask streamlines the development workflow, enabling the smooth integration of the fake job detection model into a web-based application environment.

3.3 Proposed System

The proposed system seeks to address the pervasive issue of fraudulent job postings online by leveraging machine learning techniques to develop an effective fake job detection solution. Building upon existing research, the system aims to overcome the limitations of current approaches, striving to offer a solution that is more precise, scalable, and user-friendly. Through thorough evaluation and comparison of various machine learning algorithms, including Logistic Regression, Random Forest, SVM, and Passive Aggressive models, the Passive Aggressive model emerges as the most suitable choice based on its accuracy and computational efficiency. Trained on a comprehensive dataset containing labeled examples of both genuine and fake job postings, the selected model is equipped to discern the distinguishing patterns and characteristics of fraudulent advertisements.

To optimize the performance of the machine learning model, a range of feature extraction and selection techniques are employed. Addressing the challenge of class imbalance, appropriate methods are deployed to balance the dataset and enhance the model's capability to detect the minority class (fake job postings). Furthermore, the system integrates a user-friendly interface that enables job seekers and employers to input job descriptions and receive real-time feedback on the authenticity of the job posting. Designed with simplicity and intuitiveness in mind, this interface ensures accessibility for users with diverse levels of technical proficiency. Providing clear and concise indications of a job posting's authenticity, the system empowers users to make informed decisions and steer clear of potential scams.

The proposed system not only serves as a dependable tool for job seekers to identify and avoid fake job postings but also contributes to the overall enhancement of online job platforms. By curbing the prevalence of fraudulent advertisements, the system bolsters the trust and credibility of these platforms, rendering them more dependable for both job seekers and employers alike. Additionally, the insights gleaned from developing the proposed system stand to advance the field of fake job detection research, with potential applications extending to other domains requiring similar classification tasks.

The proposed system aims to develop a fake job detection solution using machine learning techniques to effectively combat the issue of fraudulent job postings online. This system builds upon the existing research and addresses the limitations of current

approaches, aiming to provide a more accurate, scalable, and user-friendly solution. Through an extensive evaluation and comparison of various machine learning algorithms like the Logistic Regression, Random Forest, SVM, and Passive Aggressive model, the most suitable model was found to be Passive Aggressive model based on accuracy and computational efficiency. The chosen model is trained on a comprehensive dataset containing labeled examples of both genuine and fake job postings, enabling it to learn the distinguishing patterns and characteristics of fraudulent advertisements.

To enhance the performance of the machine learning model, various feature extraction and selection techniques are employed. Addressing the challenge of class imbalance, appropriate techniques are applied to balance the dataset and improve the model's ability to detect the minority class (fake job postings).

Furthermore, the proposed system incorporates a user-friendly interface that allows job seekers and employers to input job descriptions and receive real-time results indicating the authenticity of the job posting. This interface is designed with simplicity and intuitiveness in mind, ensuring accessibility to users with varying levels of technical expertise. The system will provide clear and concise indications of whether a job posting is genuine or fake, empowering users to make informed decisions and avoid potential scams.

The proposed system will not only serve as a reliable tool for job seekers to identify and avoid fake job postings but also contribute to the overall improvement of online job platforms. By reducing the prevalence of fraudulent advertisements, the system will enhance the trust and credibility of these platforms, making them more reliable for both job seekers and employers. Additionally, the insights gained from developing the proposed system will contribute to the advancement of fake job detection research and may be extended to other domains requiring similar classification tasks.

The proposed system seeks to address the pervasive issue of fraudulent job postings online by leveraging machine learning techniques to develop an effective fake job detection solution. Building upon existing research, the system aims to overcome the limitations of current approaches, striving to offer a solution that is more precise, scalable, and user-friendly. Through thorough evaluation and comparison of various machine learning algorithms, including Logistic Regression, Random Forest, SVM, and

Passive Aggressive models, the Passive Aggressive model emerges as the most suitable choice based on its accuracy and computational efficiency. Trained on a comprehensive dataset containing labeled examples of both genuine and fake job postings, the selected model is equipped to discern the distinguishing patterns and characteristics of fraudulent advertisements.

To optimize the performance of the machine learning model, a range of feature extraction and selection techniques are employed. Addressing the challenge of class imbalance, appropriate methods are deployed to balance the dataset and enhance the model's capability to detect the minority class (fake job postings). Furthermore, the system integrates a user-friendly interface that enables job seekers and employers to input job descriptions and receive real-time feedback on the authenticity of the job posting. Designed with simplicity and intuitiveness in mind, this interface ensures accessibility for users with diverse levels of technical proficiency. Providing clear and concise indications of a job posting's authenticity, the system empowers users to make informed decisions and steer clear of potential scams.

The proposed system not only serves as a dependable tool for job seekers to identify and avoid fake job postings but also contributes to the overall enhancement of online job platforms. By curbing the prevalence of fraudulent advertisements, the system bolsters the trust and credibility of these platforms, rendering them more dependable for both job seekers and employers alike. Additionally, the insights gleaned from developing the proposed system stand to advance the field of fake job detection research, with potential applications extending to other domains requiring similar classification tasks.

3.4 Diagrams

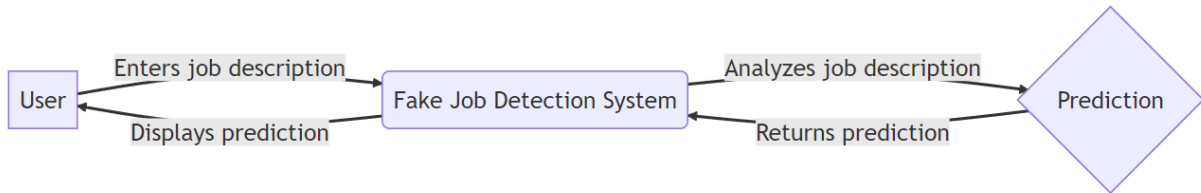


Figure 1: Use Case Diagram

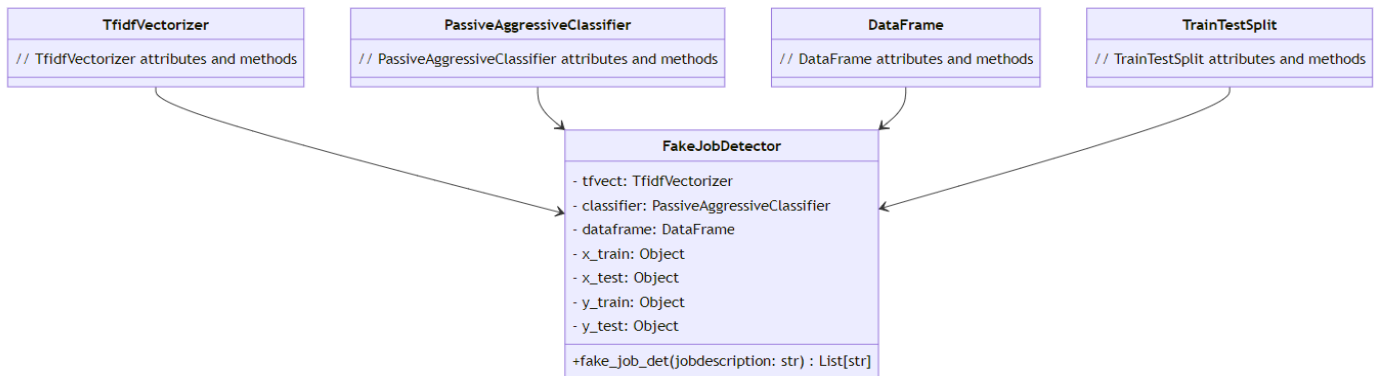


Figure 2: Class Diagram

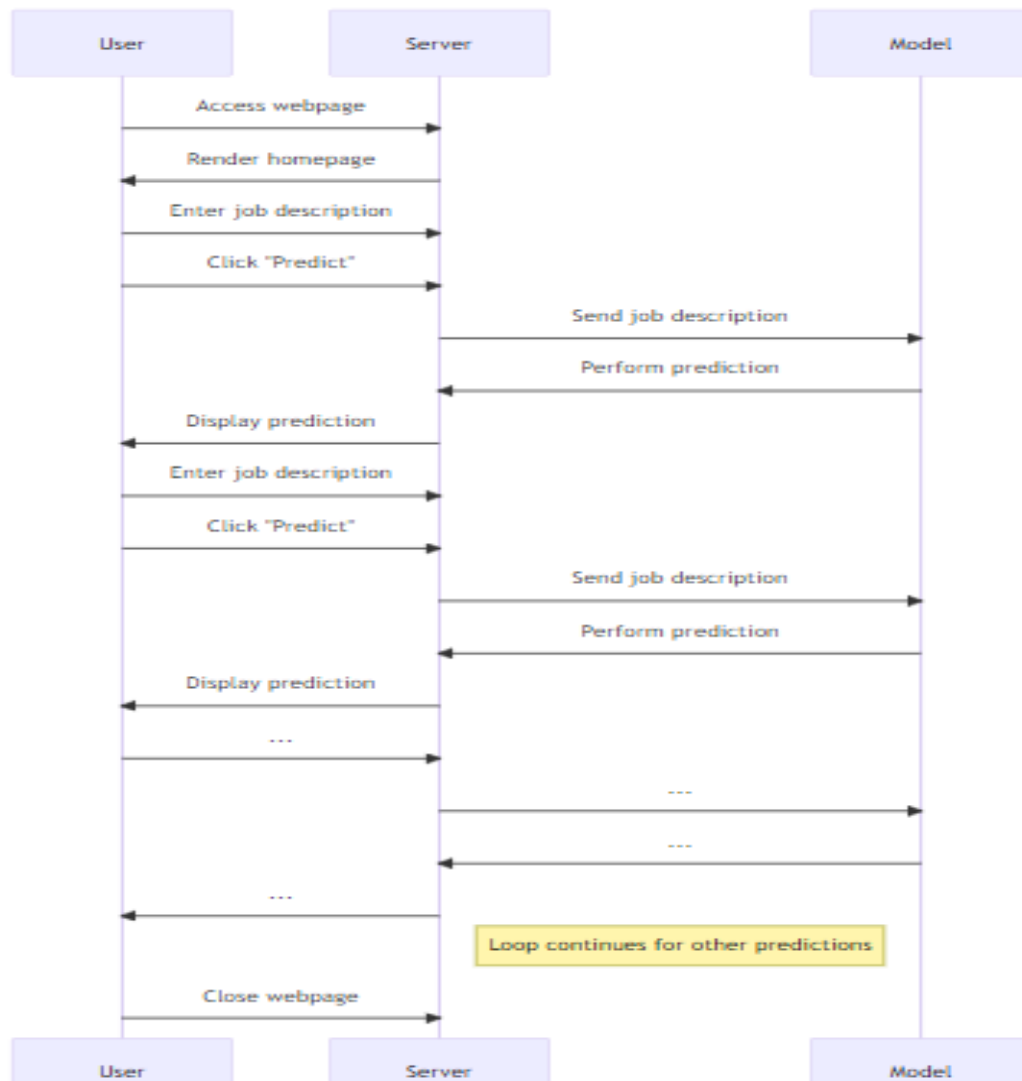


Figure 3: Sequence Diagram

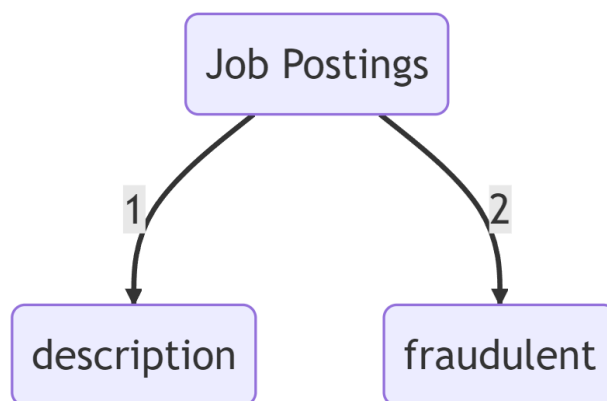


Figure 4: Schema Diagram

3.5 Description of the software used

3.5.1. The Front-End User Interface

The user interface is developed using HTML and CSS. It provides a visually appealing and interactive platform for users to input job descriptions and receive the detection results. The user interface collects the job description text entered by the user and sends it to the model for processing with the help of Flask.

3.5.2. Flask Server

The Flask server acts as the backbone of the system, handling the communication between the user interface, and machine learning model. Upon receiving the job description input, the server passes it to the machine learning model for preprocessing using Flask's routing capabilities. The preprocessed text is then fed into the machine learning model for prediction. Once the prediction is obtained, the server sends the result back to the user interface for display.

3.5.3. Feature Extraction with TF-IDF

The system employs the TF-IDF (Term Frequency-Inverse Document Frequency) technique as part of the feature extraction process. This technique quantifies the importance of each word in the job description by considering both its frequency within the document and its rarity across all documents. TF-IDF helps in capturing the distinguishing characteristics of the job descriptions, allowing the system to make informed predictions.

3.5.4. The Machine Learning Model

The machine learning model used in the system is the Passive Aggressive Classifier. This classifier is known for its effectiveness in handling binary classification tasks, making it suitable for distinguishing between genuine and fake job postings. The model is trained using labeled data, where genuine and fake job postings are used to learn patterns and make predictions on unseen data.

3.6 Code Snippets

3.6.1 HTML

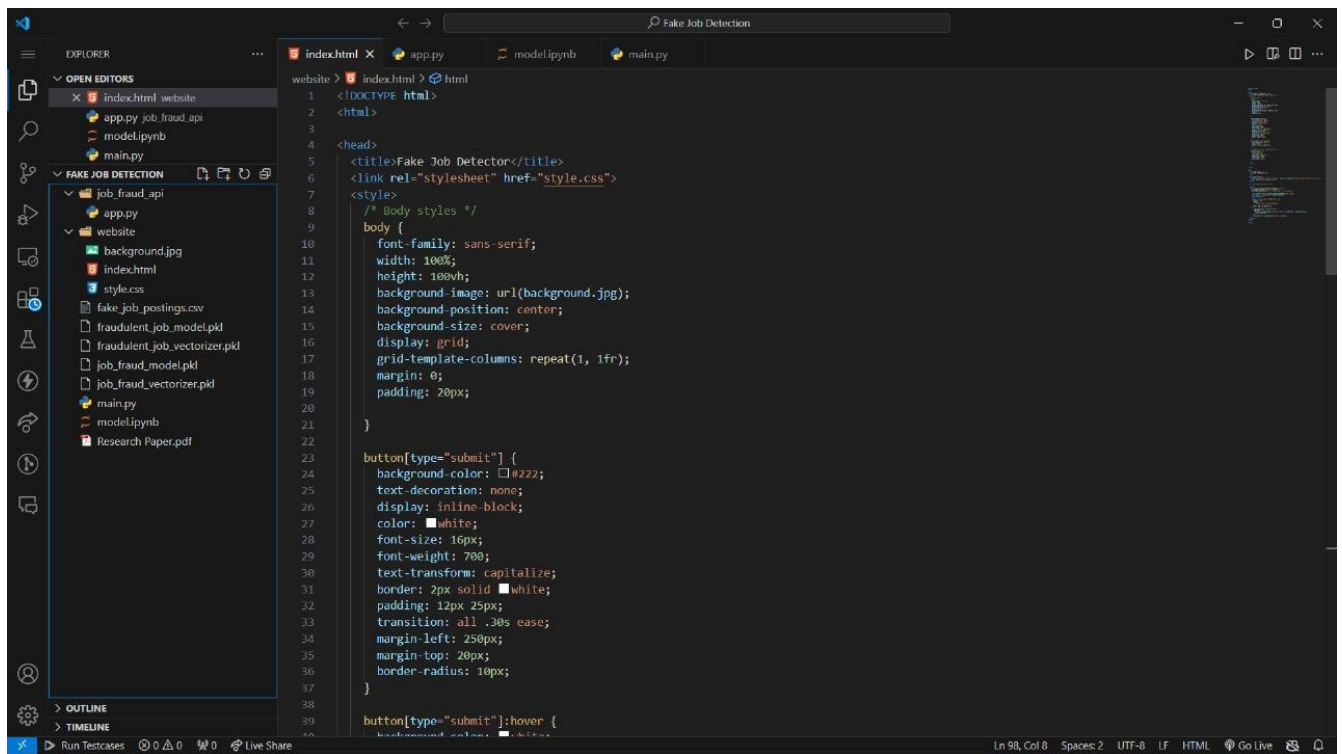


Figure 5

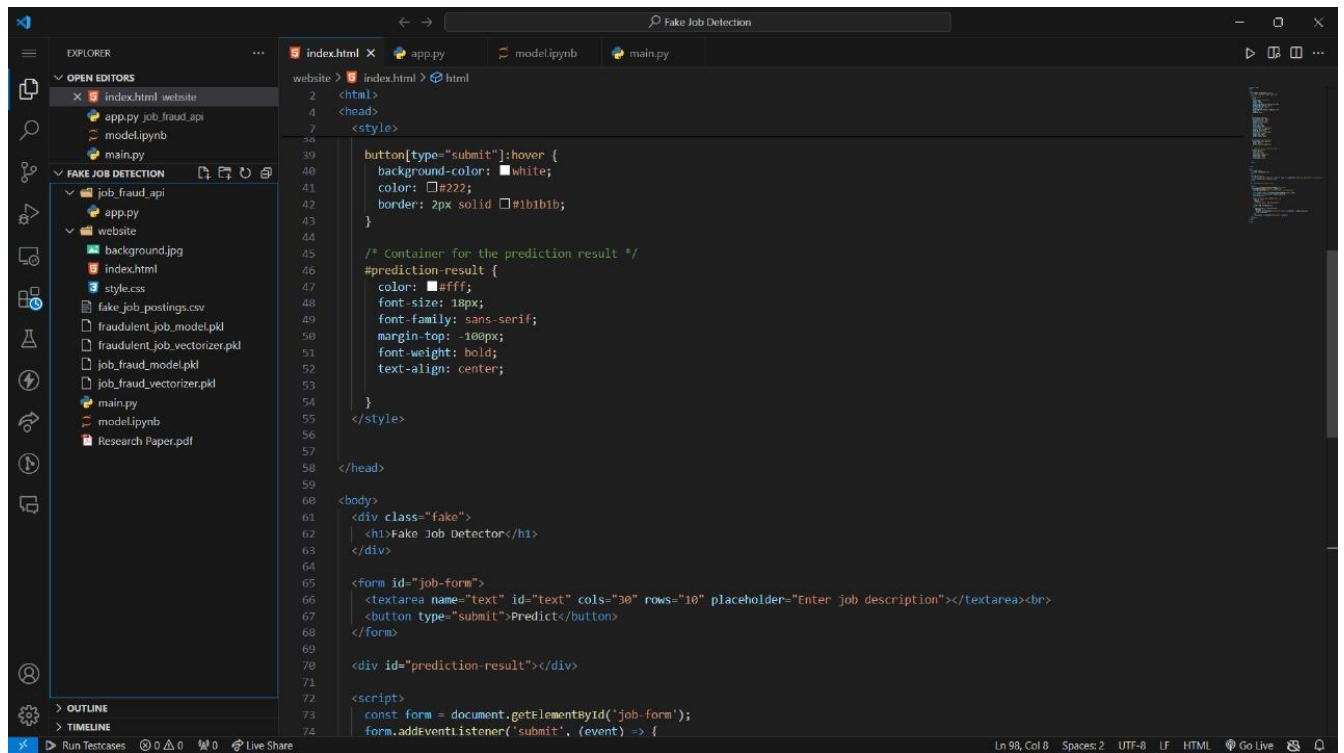


Figure 6

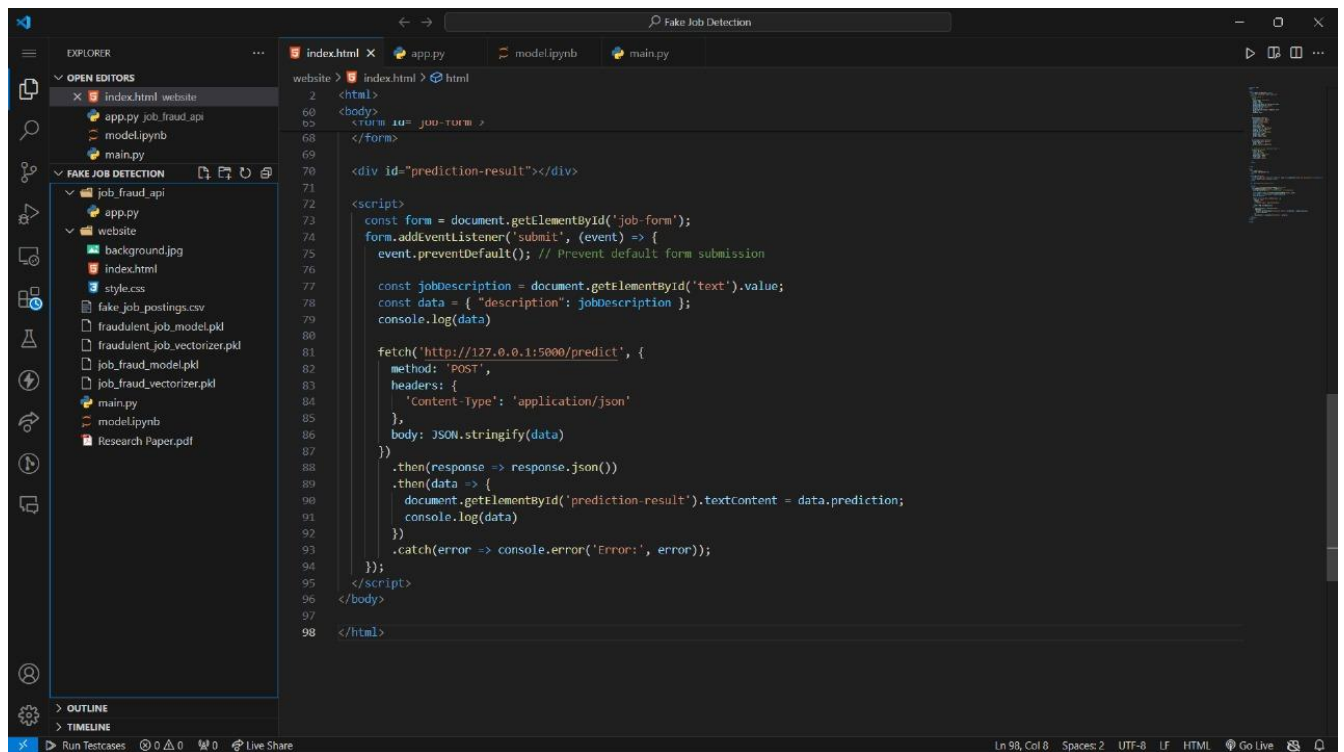


Figure 7

3.6.2 CSS

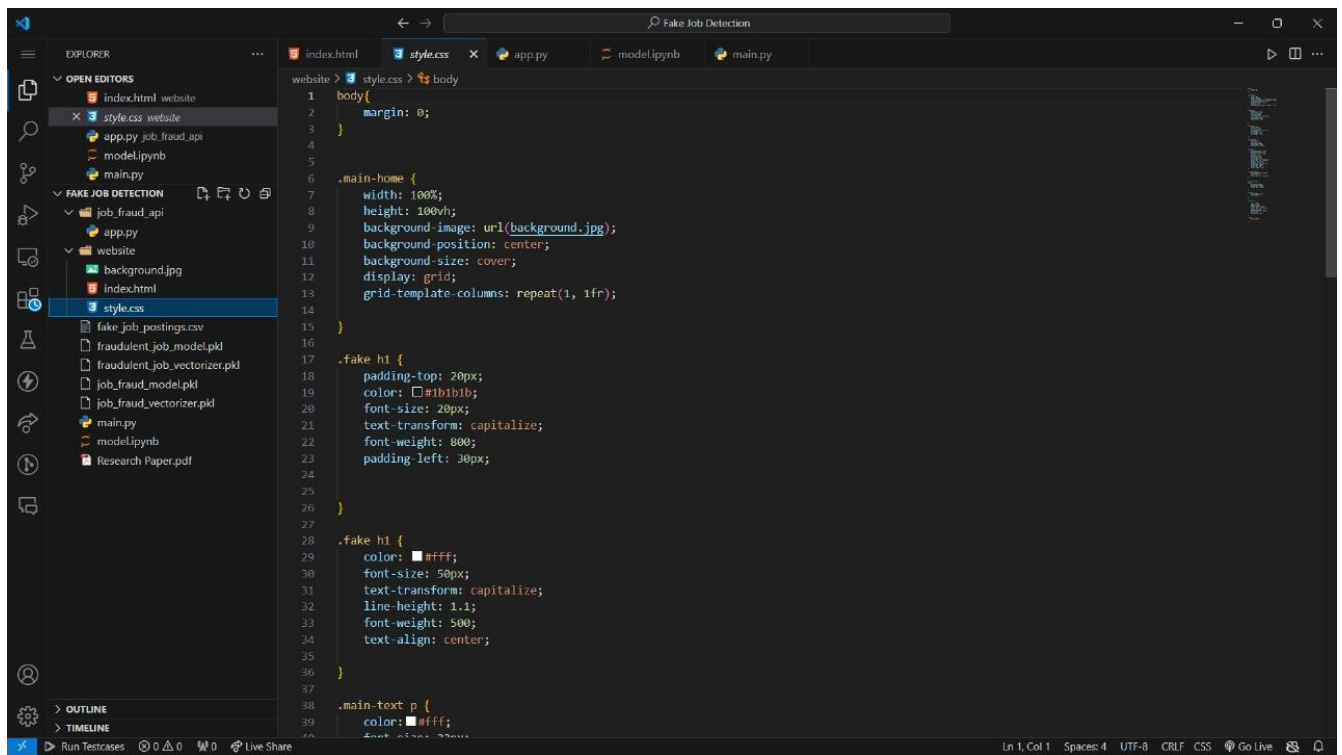


Figure 8

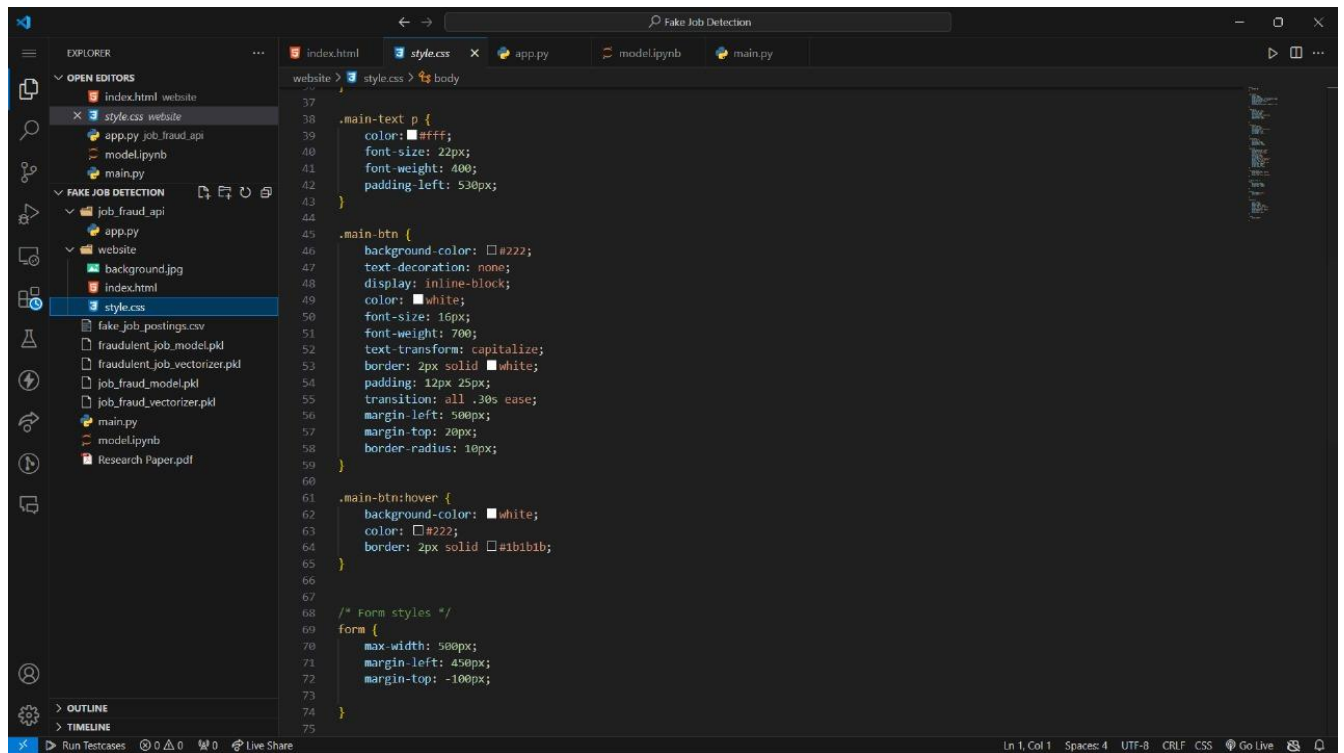


Figure 9

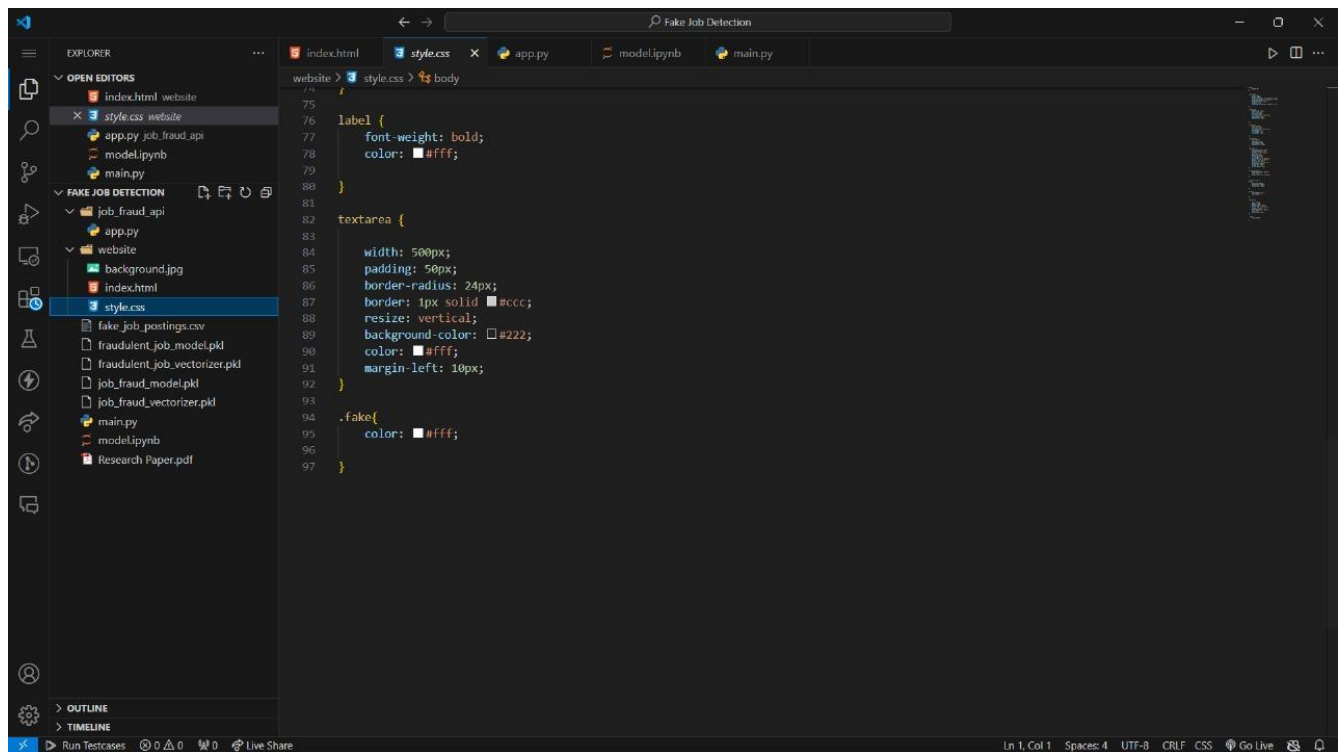


Figure 10

3.6.3 API-Flask

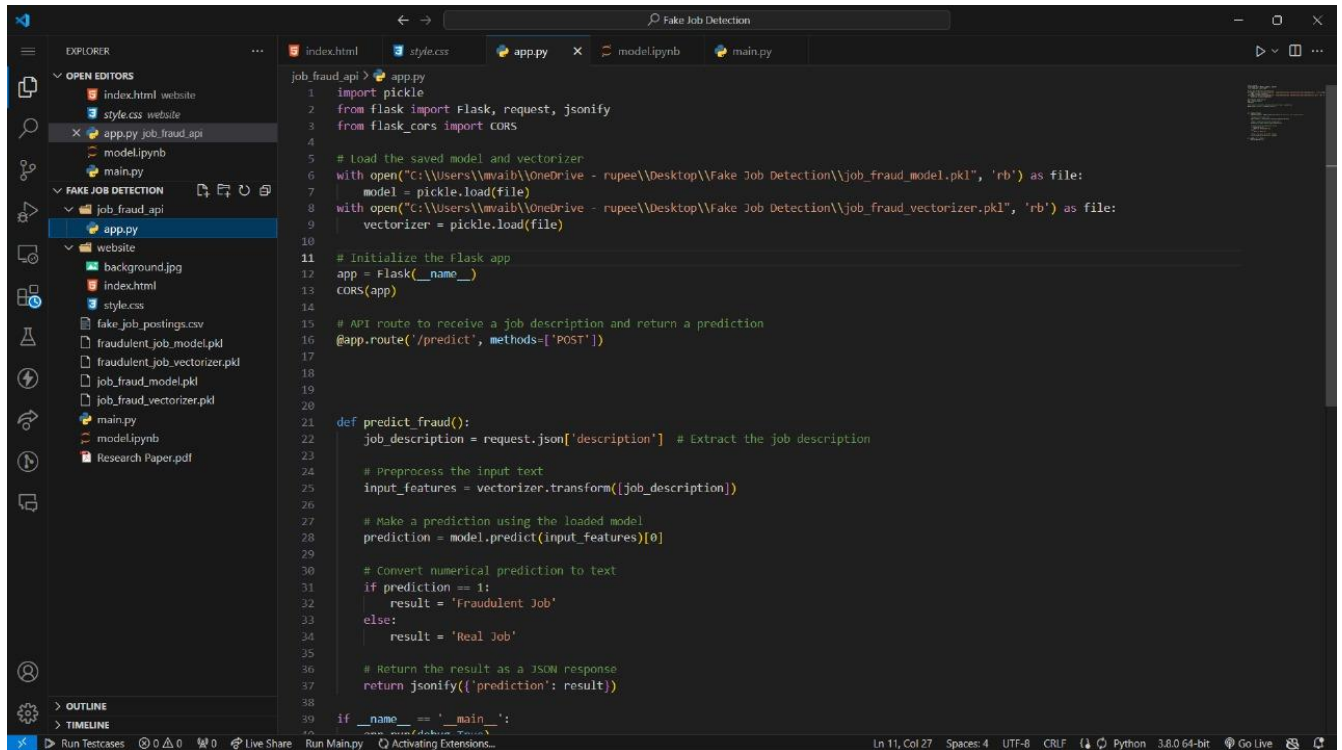


Figure 11

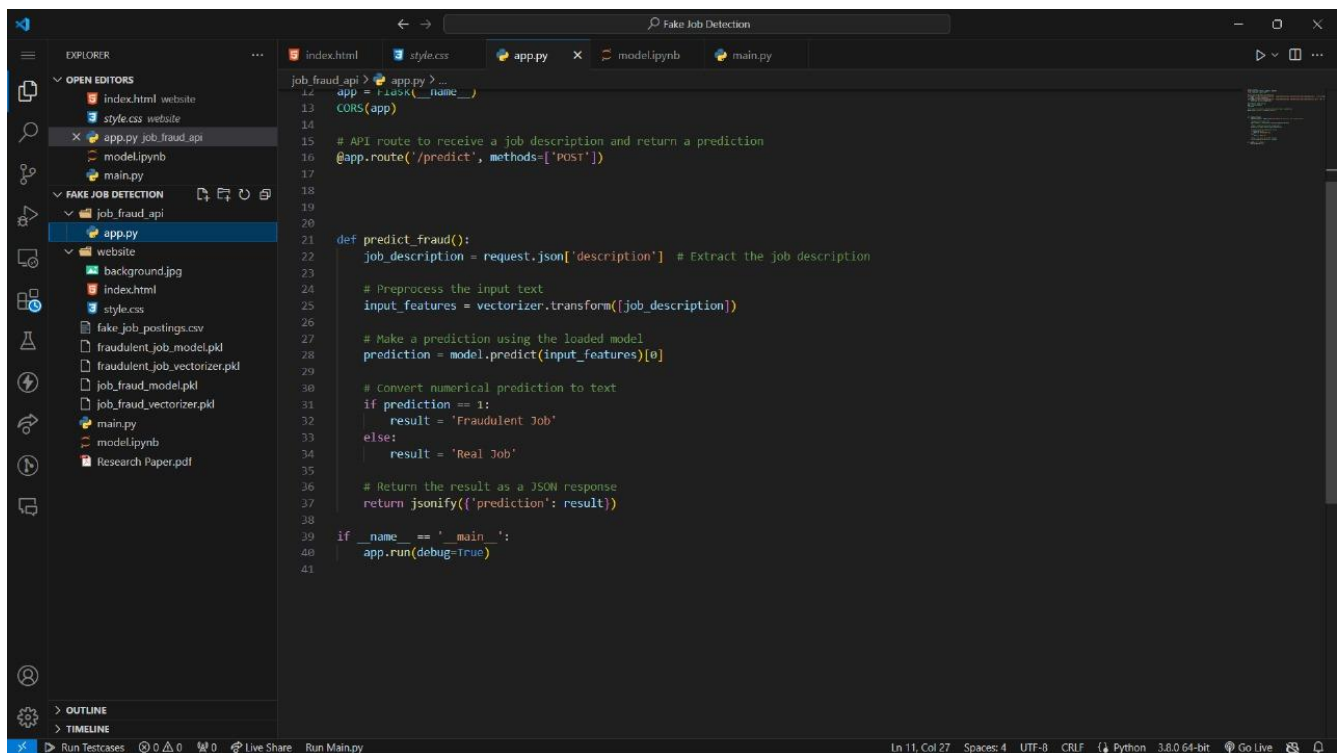


Figure 12

3.6.4 Python

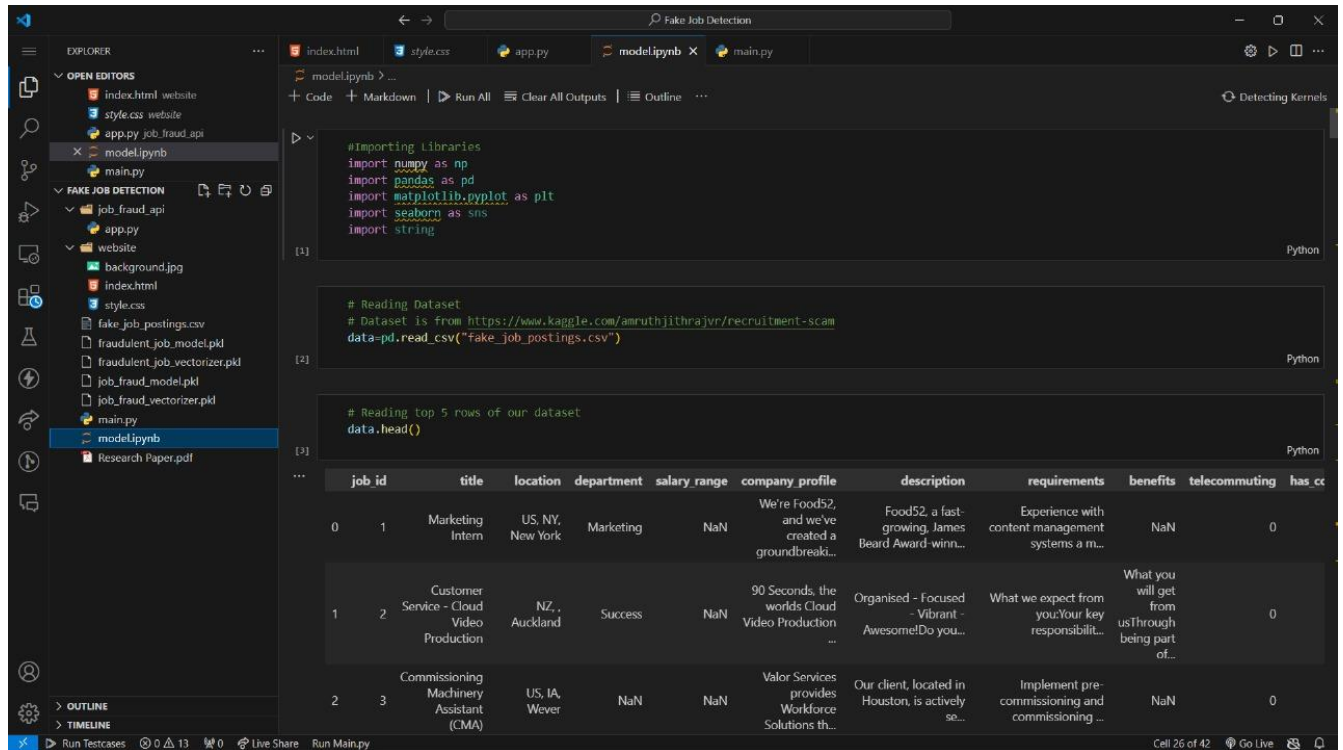


Figure 13

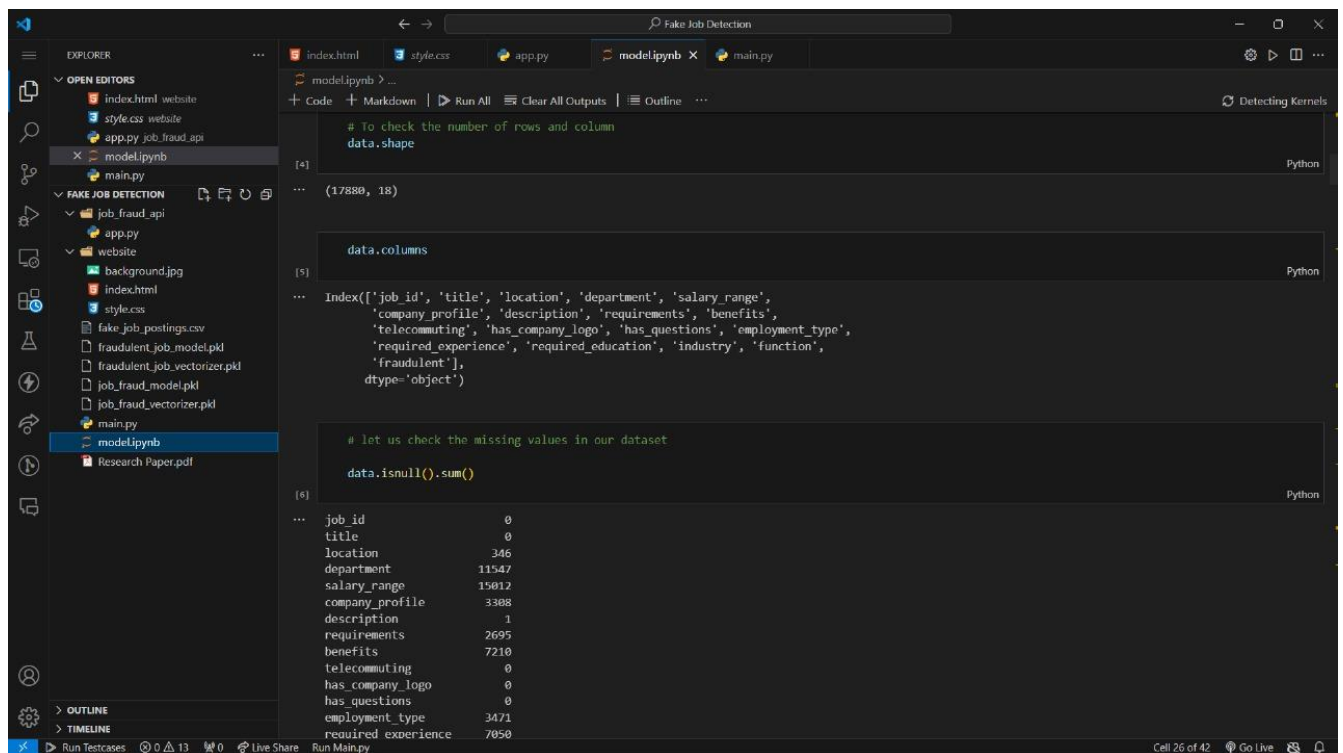


Figure 14

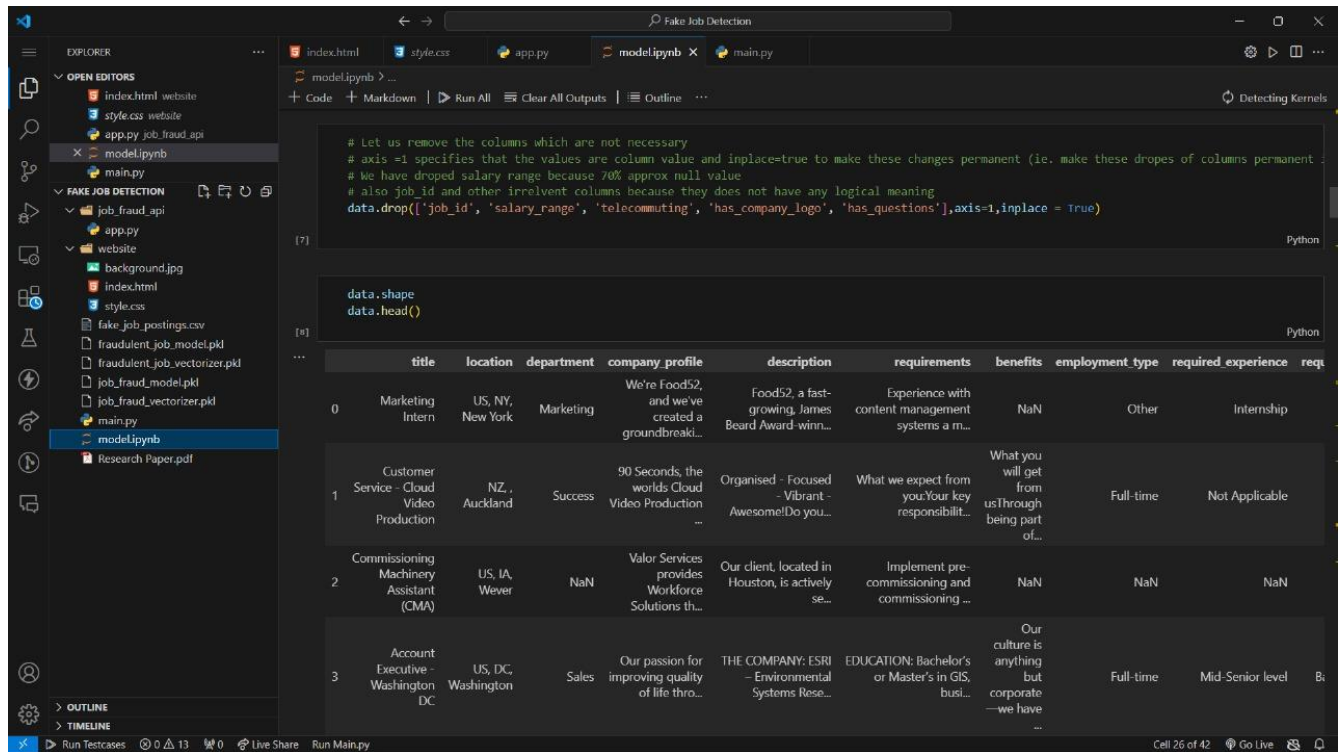


Figure 15

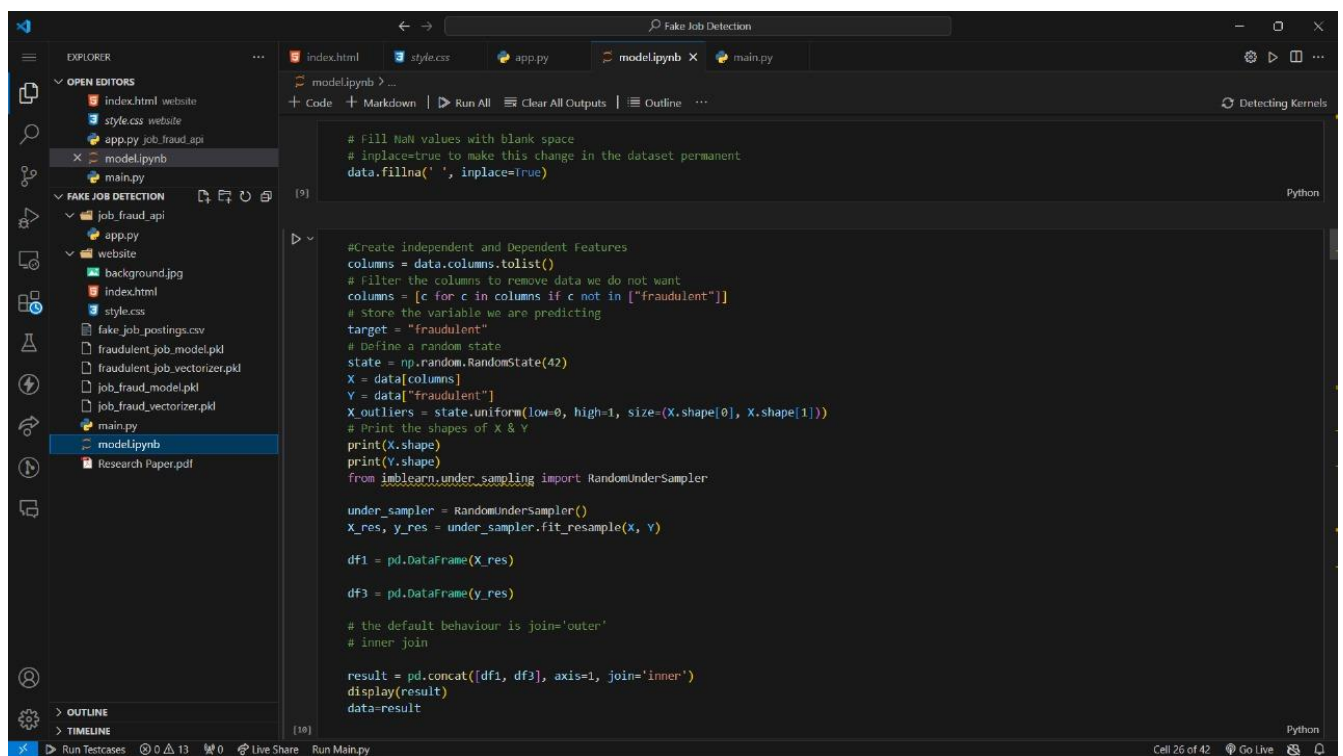


Figure 16

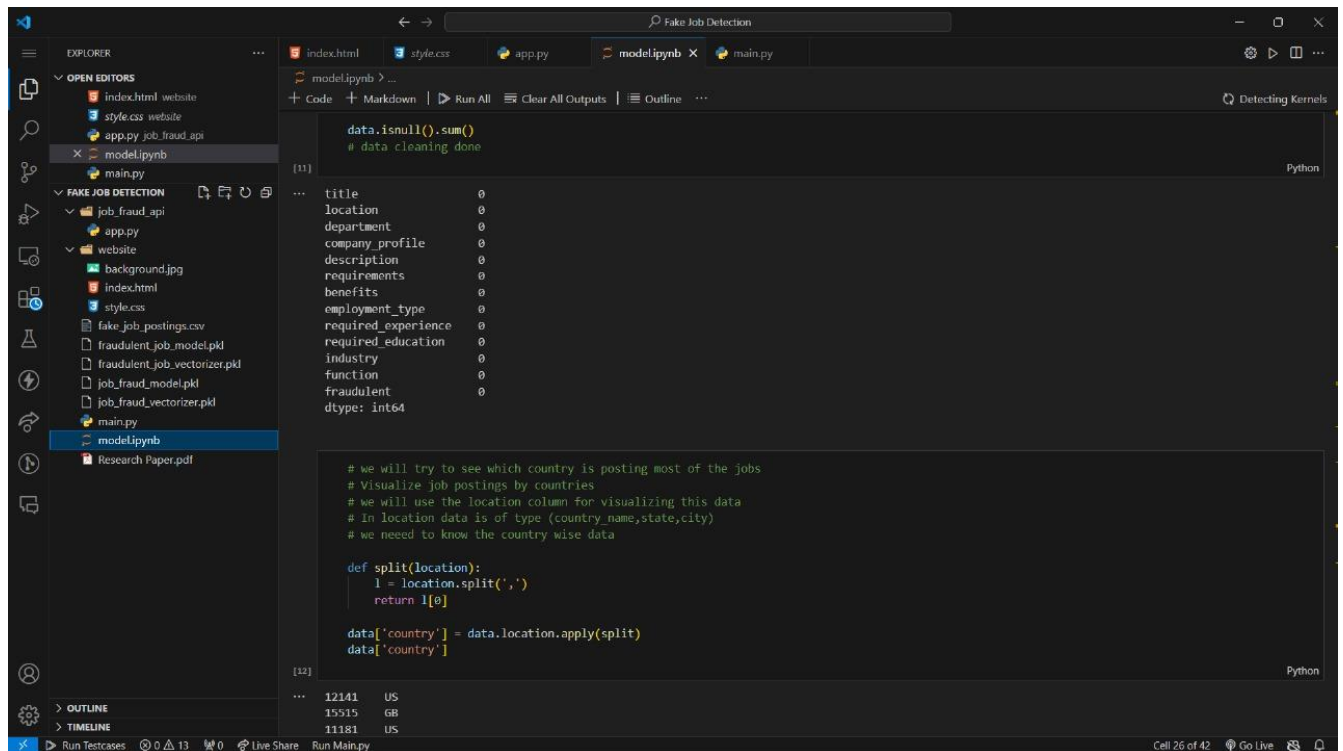


Figure 17

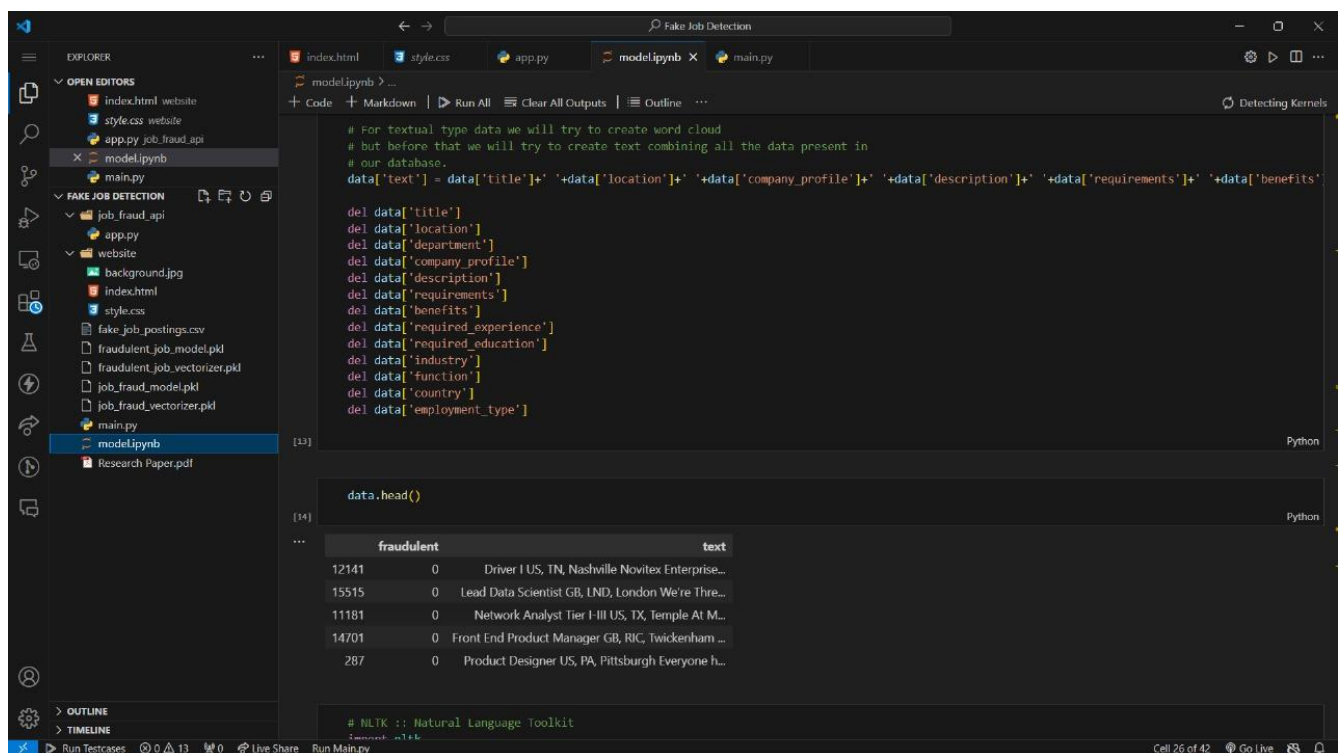


Figure 18

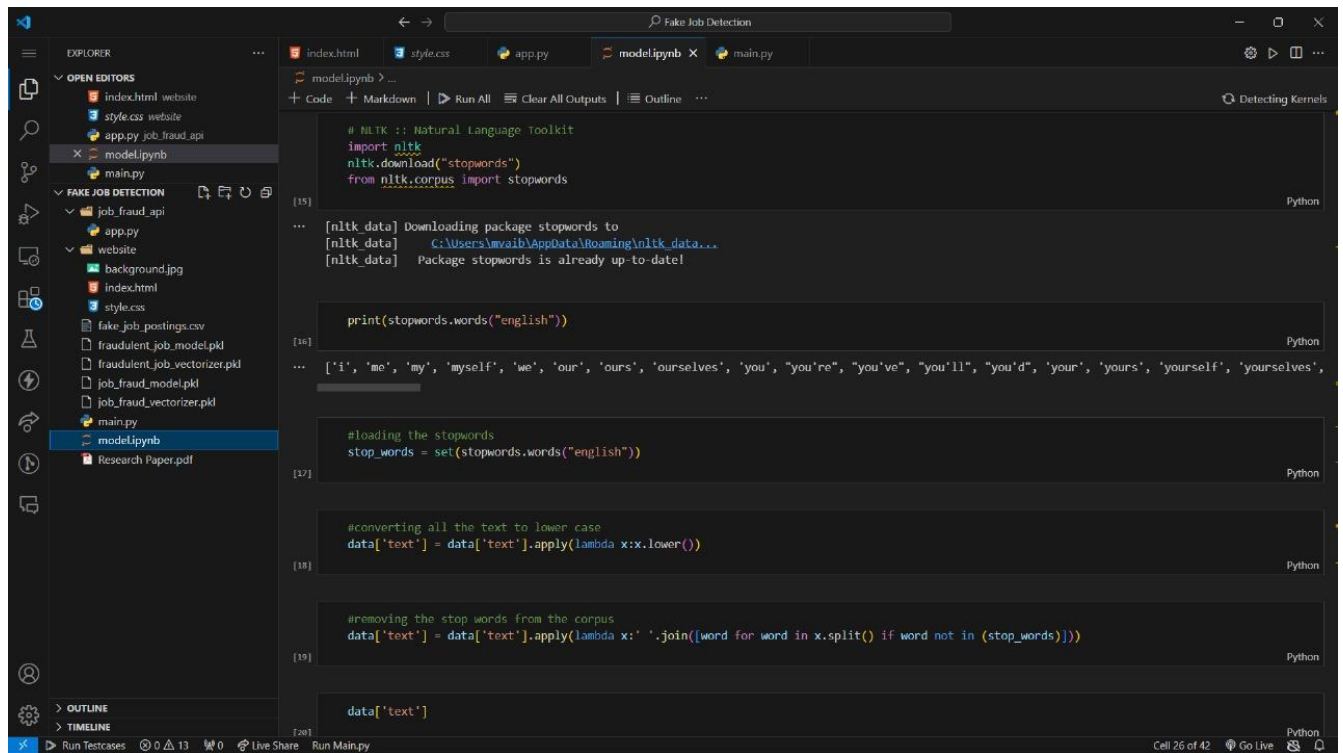


Figure 19

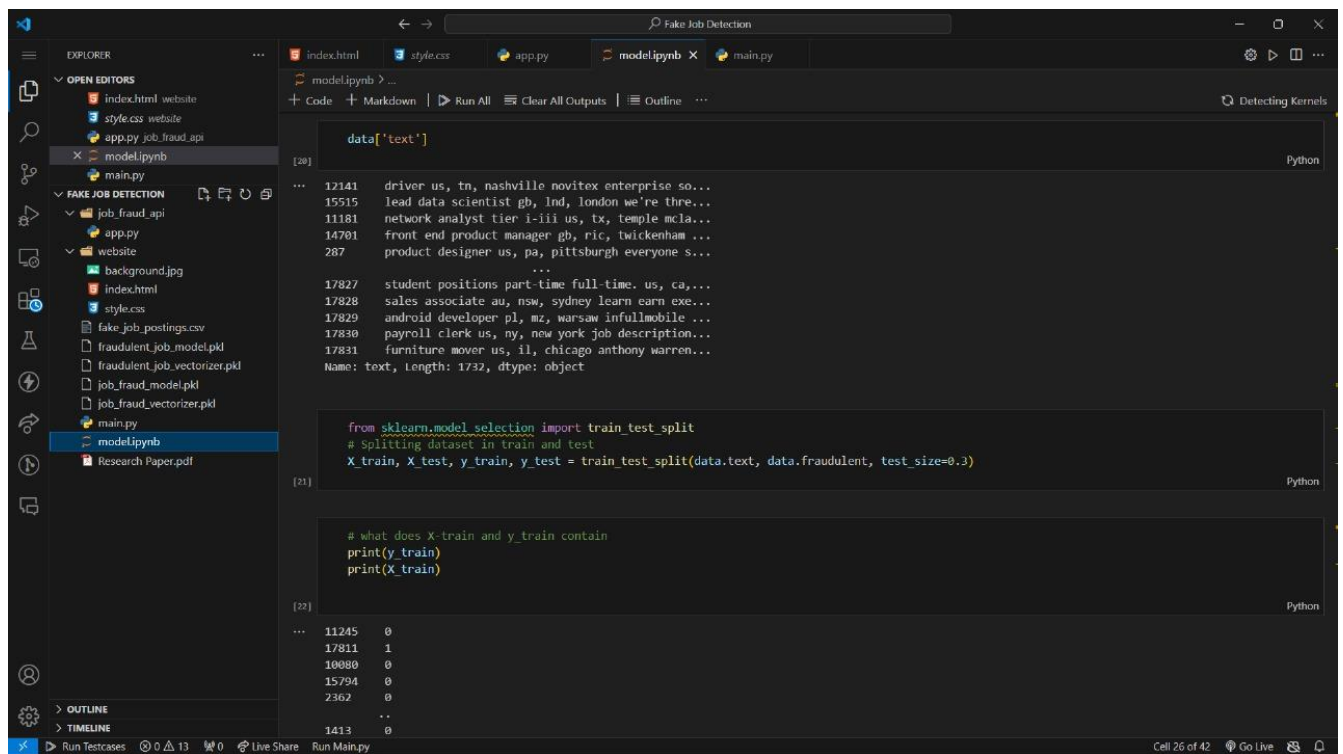


Figure 20

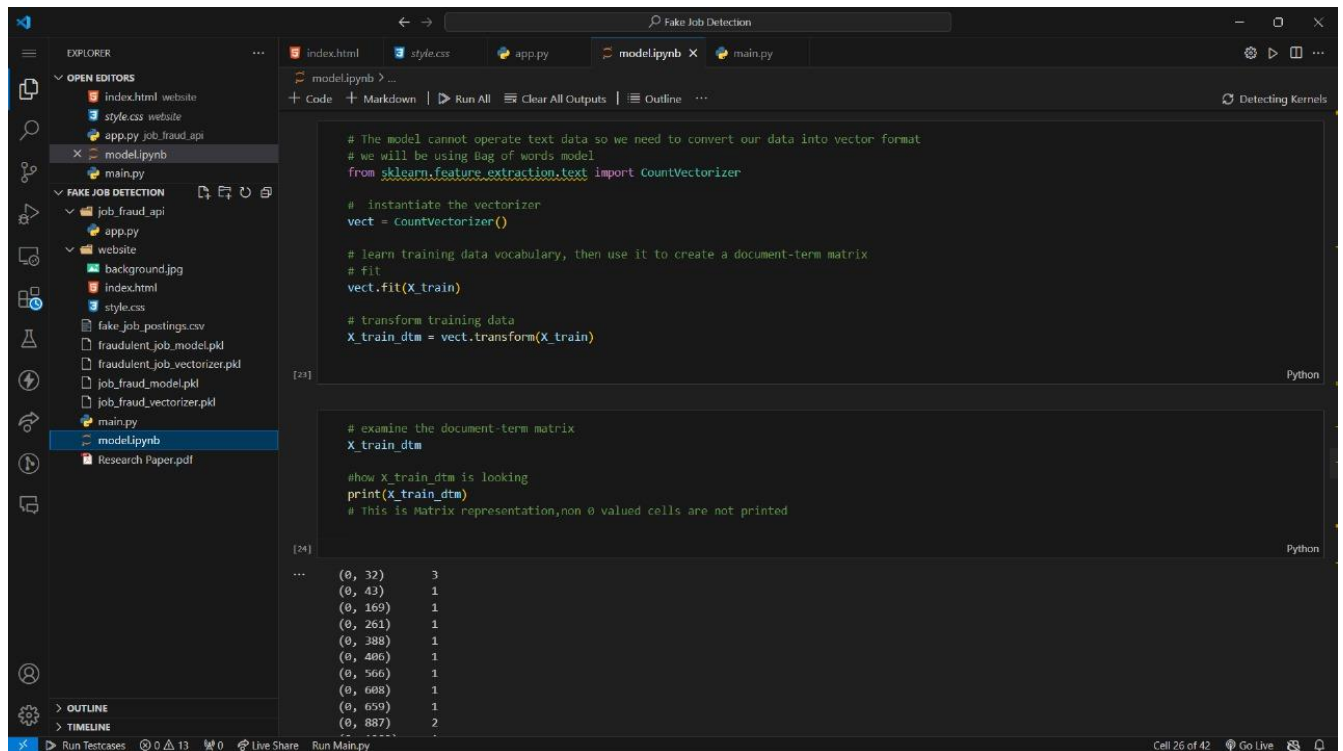


Figure 21

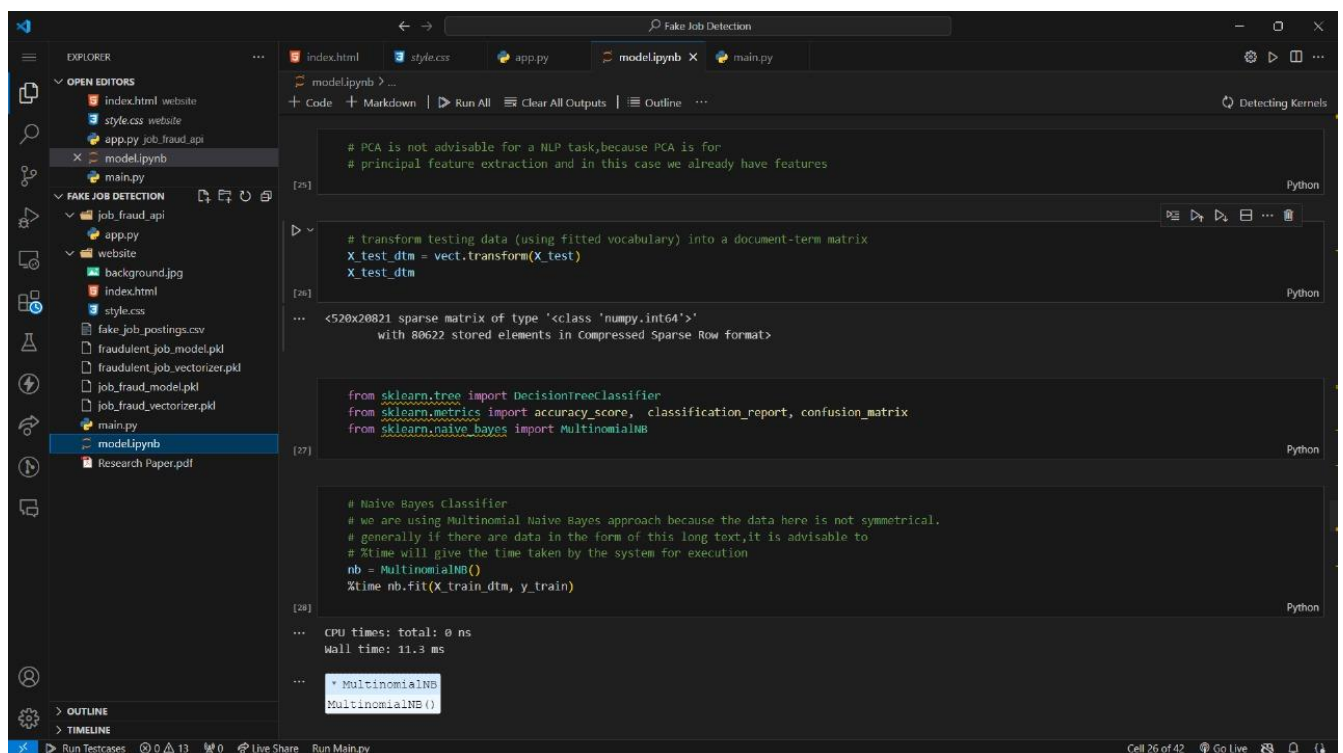


Figure 22

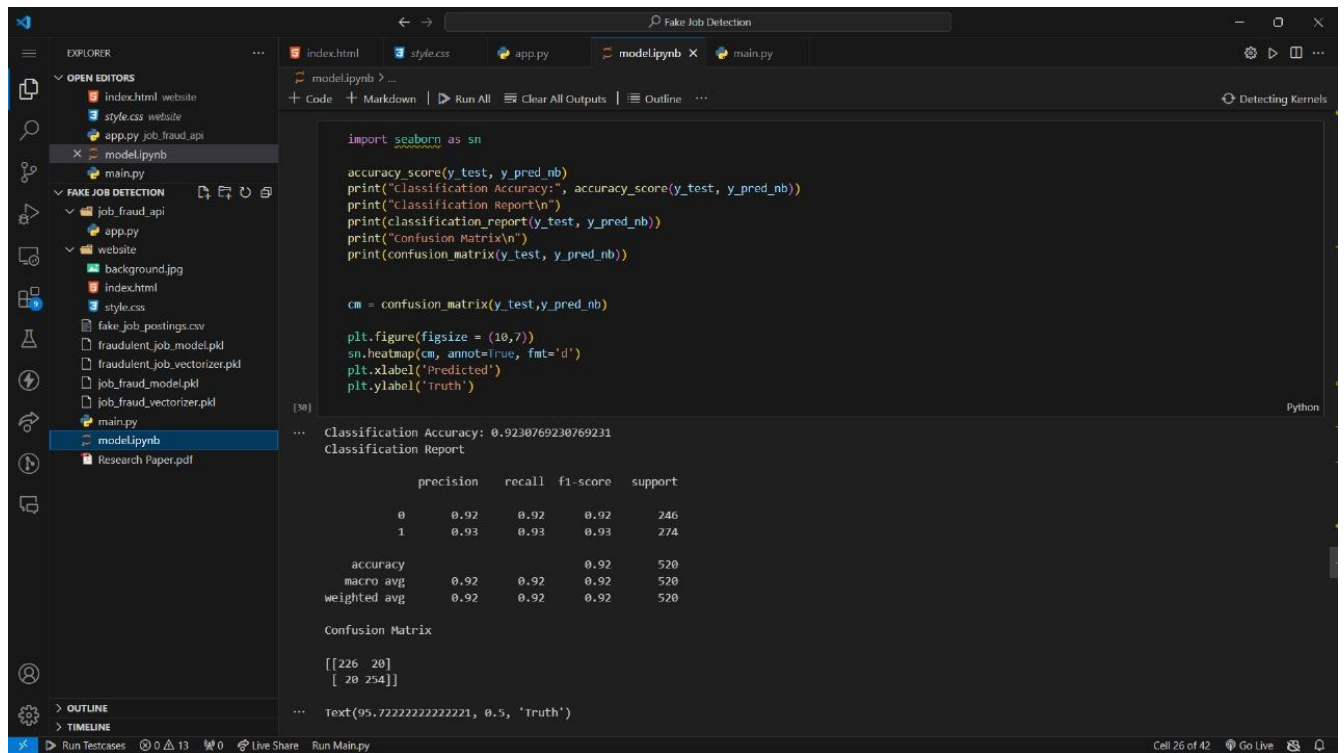


Figure 23

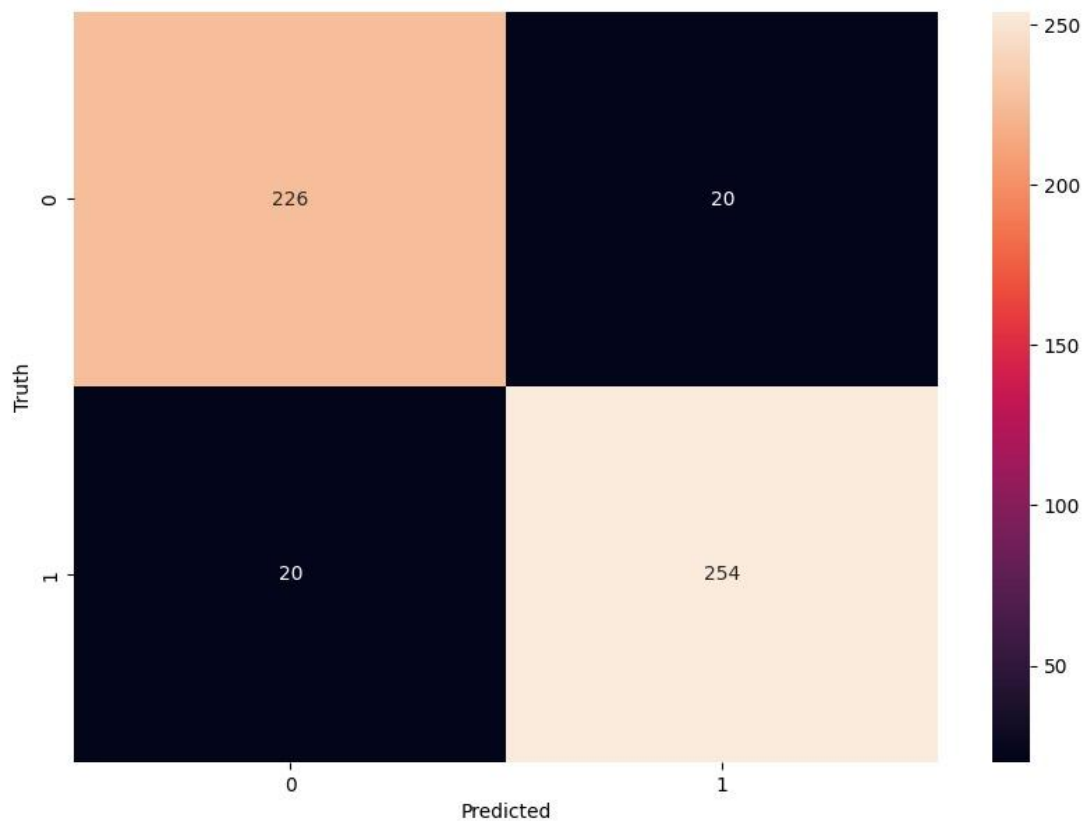


Figure 24

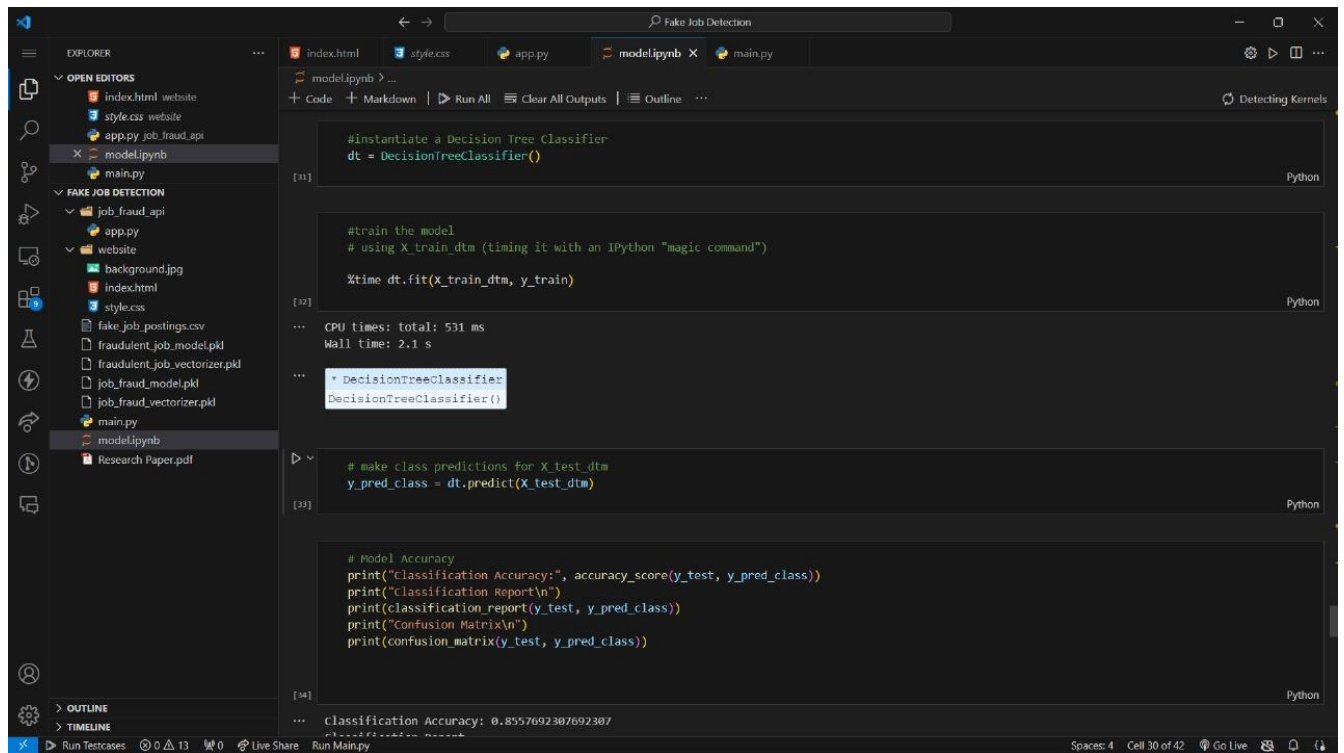


Figure 25

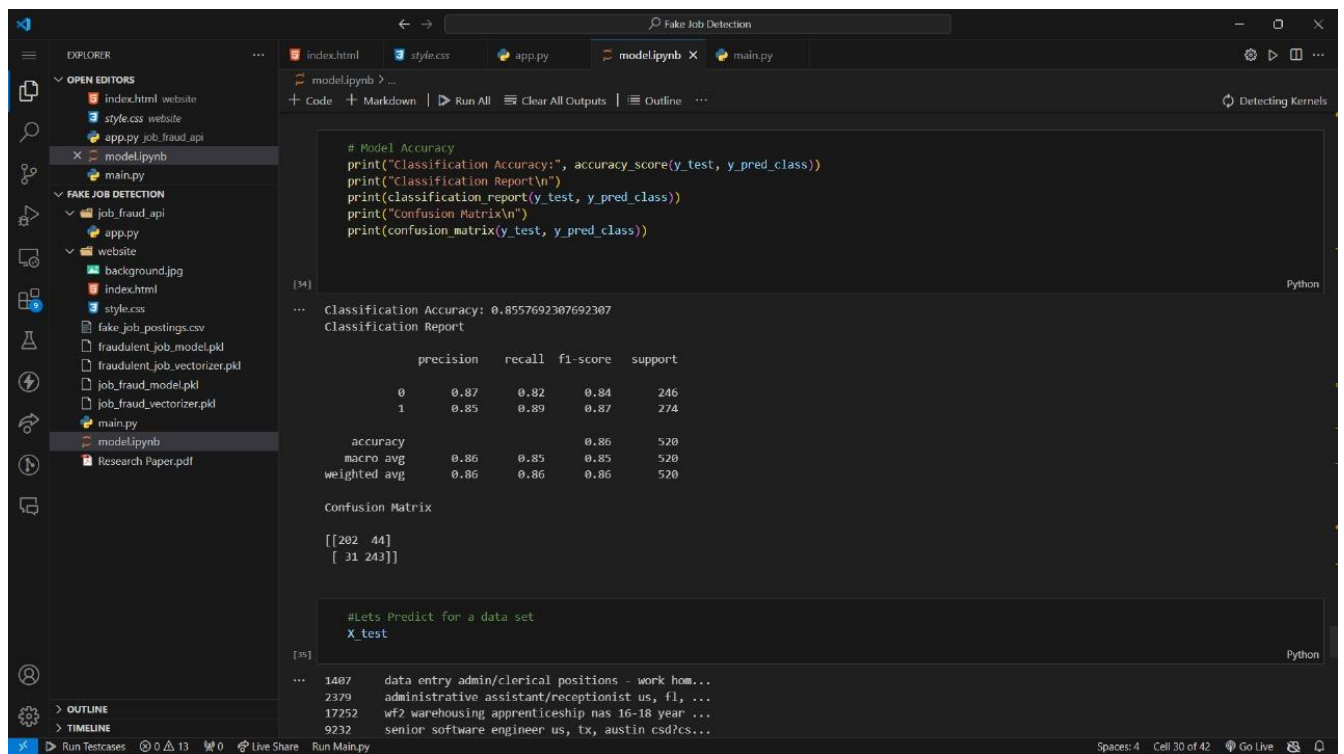


Figure 26

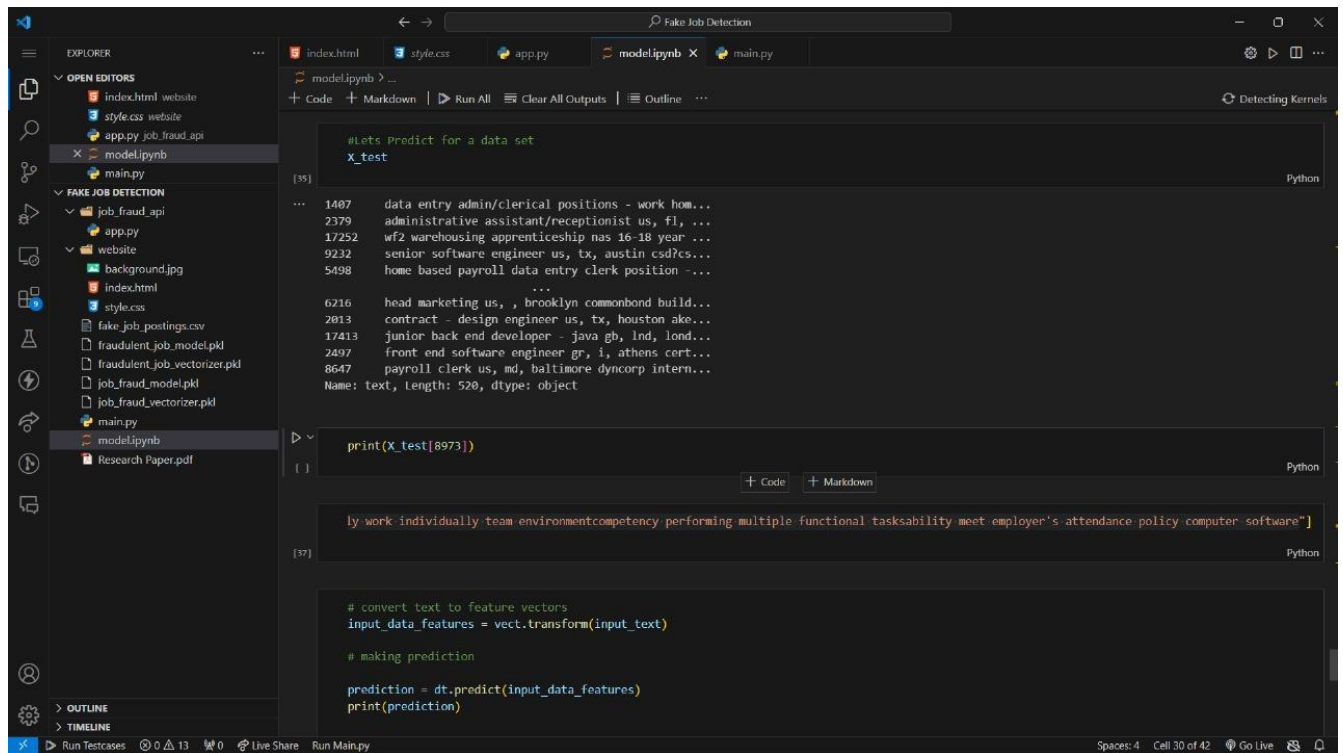


Figure 27

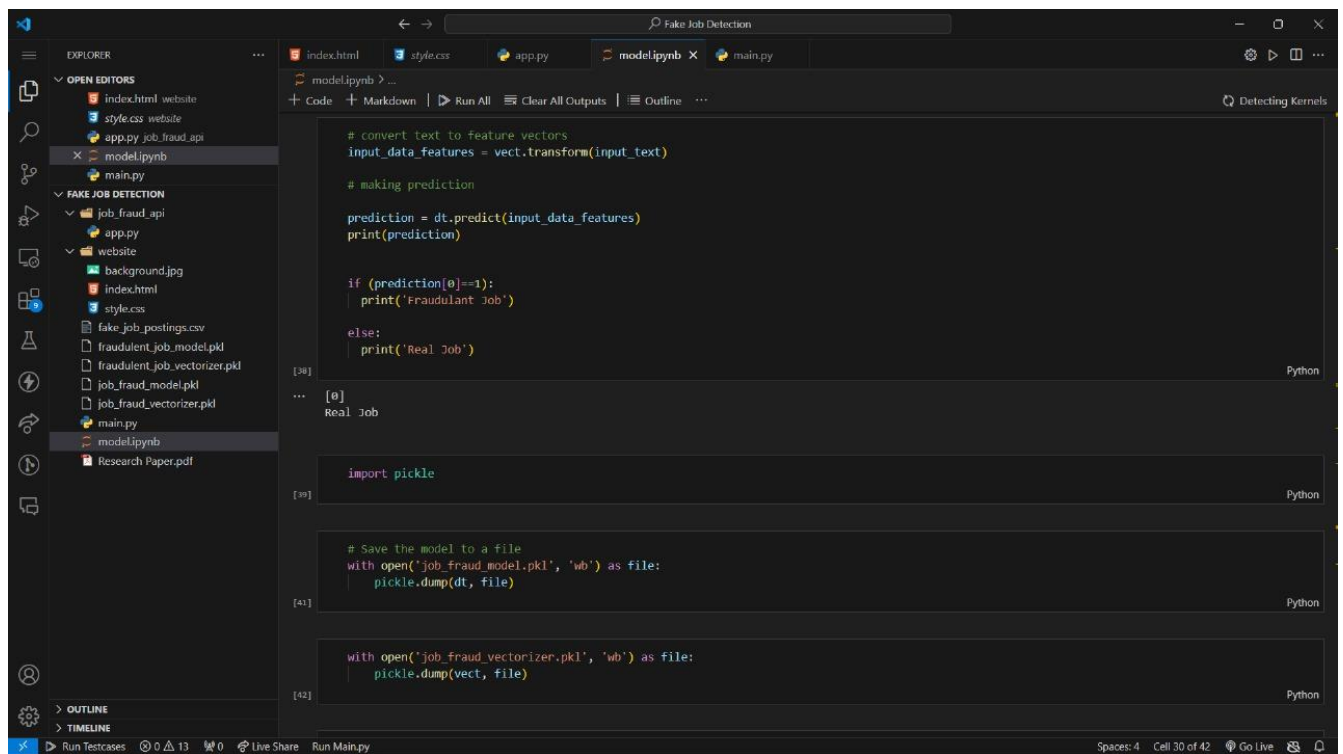


Figure 28

CHAPTER 4

4 RESULT ANALYSIS AND VALIDATION

4.1. Experimental Setup

4.1.1. Literature Review

The journey of experimentation commenced with a meticulous literature review, delving into the depths of seven seminal research papers centered on fake job detection through machine learning methodologies. Each of these papers underwent thorough scrutiny, offering invaluable perspectives on prevailing approaches, algorithms, and techniques prevalent within the domain.

The foundation of the experimental endeavor was laid with an exhaustive literature review, encompassing a comprehensive analysis of seven seminal research papers dedicated to the realm of fake job detection utilizing machine learning paradigms. Through meticulous examination, these papers offered profound insights into the diverse array of approaches, algorithms, and techniques prevalent within the field.

Embarking on the journey of experimentation, a critical literature review served as the cornerstone, entailing a thorough examination of seven seminal research papers focused on the application of machine learning in fake job detection. Delving deep into the nuances of each paper, invaluable insights were gleaned, shedding light on the myriad approaches, algorithms, and techniques prevalent within the domain.

4.1.2. Model Selection

The process of model selection commenced with the evaluation of various machine learning algorithms, encompassing logistic regression, SVM, random forest, and the passive-aggressive classifier. Through rigorous assessment, the aim was to identify the optimal model for fake job detection. Utilizing a preprocessed dataset, each algorithm underwent meticulous training, followed by a thorough comparison of their performances using pertinent evaluation metrics, with accuracy serving as a primary benchmark.

In the quest for the most suitable model for fake job detection, an extensive evaluation of multiple machine learning algorithms was undertaken. Logistic regression, SVM, random forest, and the passive-aggressive classifier were among the contenders subjected to rigorous scrutiny. Leveraging a preprocessed dataset, each algorithm underwent systematic training, paving the way for a comprehensive comparison of their performances using relevant evaluation metrics. Accuracy emerged as the primary yardstick in determining the efficacy of each model.

In the pursuit of identifying the optimal model for fake job detection, a meticulous process of model selection unfolded. Several machine learning algorithms, including logistic regression, SVM, random forest, and the passive-aggressive classifier, underwent thorough evaluation. With a preprocessed dataset at hand, each algorithm was meticulously trained, setting the stage for a meticulous comparison of their performances using appropriate evaluation metrics. Throughout this process, accuracy served as a crucial metric in gauging the effectiveness of each model.

4.1.3. Data Collection

For the successful execution of the experiments, a pivotal aspect was the acquisition of an appropriate dataset comprising job postings. To fulfill this requirement, an exhaustive compilation of both genuine and fake job postings was procured from Kaggle. Particular attention was paid to curating a dataset that was not only diverse and representative but also sufficiently large to generate dependable results.

A crucial prerequisite for conducting the experiments was the procurement of a suitable dataset comprising job postings. This necessitated a meticulous collection process, whereby a comprehensive array of both genuine and fake job postings was sourced from Kaggle. Emphasis was placed on assembling a dataset that exhibited diversity, representativeness, and substantial volume, ensuring the reliability of the ensuing results.

The foundation of the experiments rested upon the acquisition of a pertinent dataset consisting of job postings. This entailed a thorough collection process, whereby a wide-ranging assortment of both genuine and fake job postings was sourced from Kaggle. Special attention was dedicated to curating a dataset that not only encompassed diversity and representativeness but also boasted a considerable size, thereby instilling confidence in the reliability of the outcomes.

4.1.4. Data Preprocessing

Before proceeding with the analysis, it was imperative to subject the collected dataset to preprocessing steps aimed at cleansing and refining the data. This entailed the meticulous removal of irrelevant or noisy data, along with the systematic handling of missing values. Additionally, efforts were directed towards rectifying any inconsistencies or errors present within the dataset, ensuring its integrity and reliability for subsequent analysis.

The integrity of the collected dataset was paramount, necessitating a series of preprocessing steps to prepare the data for analysis. This involved the systematic removal of irrelevant or noisy data, meticulous handling of missing values, and diligent rectification of any inconsistencies or errors within the dataset. By undertaking these measures, the dataset was refined and primed for accurate analysis and interpretation.

To lay the groundwork for meaningful analysis, the collected dataset underwent rigorous preprocessing steps. These crucial procedures encompassed the removal of irrelevant or noisy data, meticulous handling of missing values, and diligent rectification of any inconsistencies or errors present within the dataset. Through these meticulous efforts, the dataset was meticulously refined and prepared for subsequent analysis, ensuring the integrity and reliability of the findings.

4.1.5. Feature Extraction

Following the preprocessing stage, the next crucial step involved extracting relevant features from the processed job descriptions. In this project, the TF-IDF (Term Frequency-Inverse Document Frequency) technique was leveraged to convert the textual data into a numerical representation. By employing TF-IDF, the significance of each word in the job descriptions was captured, thereby facilitating the creation of meaningful inputs for the machine learning models.

With the job descriptions processed, the subsequent task entailed extracting pertinent features to serve as inputs for the machine learning models. For this purpose, the TF-IDF (Term Frequency-Inverse Document Frequency) technique was enlisted, enabling the transformation of textual data into a numerical representation. Through TF-IDF, the relative importance of each word within the job descriptions was quantified, thus furnishing the models with meaningful numerical inputs.

Having completed the preprocessing phase, the focus shifted towards extracting meaningful features from the processed job descriptions. To achieve this, the TF-IDF (Term Frequency-Inverse Document Frequency) technique was harnessed, facilitating the conversion of textual data into a numerical representation. By leveraging TF-IDF, the relevance of each word within the job descriptions was quantified, thereby enabling the generation of informative numerical features for the machine learning models.

4.1.6. Model Training and Validation

The chosen machine learning model, the passive-aggressive classifier, underwent rigorous training and validation procedures using the preprocessed dataset. To gauge the model's performance accurately, the dataset was divided into training and testing subsets. Through this meticulous process, the efficacy and reliability of the model were thoroughly assessed.

To ensure the robustness of the selected machine learning model, namely the passive-aggressive classifier, it underwent comprehensive training and validation processes using the preprocessed dataset. By partitioning the dataset into distinct training and testing subsets, the model's performance was meticulously evaluated. Through this systematic approach, insights into the model's effectiveness and generalization capabilities were garnered.

With the aim of assessing the performance and reliability of the chosen machine learning model, the passive-aggressive classifier, rigorous training and validation procedures were conducted using the preprocessed dataset. Through the strategic division of the dataset into training and testing subsets, the model's ability to generalize and perform accurately was rigorously evaluated. This meticulous process yielded valuable insights into the model's efficacy and suitability for the intended task.

4.1.7. Result Analysis

The evaluation of the trained model's performance centered on the analysis of prediction accuracy. Through this process, the model's efficacy in accurately detecting fake job postings was assessed and juxtaposed against the findings reported in the existing literature.

A critical aspect of assessing the trained model's performance involved conducting a thorough analysis of prediction accuracy. This entailed evaluating the model's proficiency in accurately discerning fake job postings and juxtaposing the obtained results with those documented in the literature.

The evaluation of the trained model's performance hinged on the meticulous analysis of prediction accuracy. Through this process, the model's ability to effectively identify fake job postings was rigorously scrutinized and compared against the benchmarks established in the existing literature.

4.1.8. Experimental Validation

To ascertain the effectiveness of the proposed system, rigorous experimental validation was conducted. The final model and experimental setup were put to the test using real-world job postings. This critical validation step was instrumental in ensuring that the system's performance transcended theoretical boundaries and aligned with real-world scenarios.

The validation of the proposed system's efficacy was substantiated through a robust experimental validation process. By subjecting the final model and experimental setup to real-world job postings, the system's performance was rigorously evaluated. This pivotal validation step served as a litmus test, ensuring that the system's capabilities extended beyond mere theoretical constructs and were reflective of practical utility.

The validation of the proposed system's effectiveness was fortified through meticulous experimental validation efforts. Leveraging real-world job postings, the final model and experimental setup underwent rigorous testing to evaluate the system's performance. This essential validation step served as a testament to the system's practical applicability, affirming its efficacy in real-world contexts.

4.2. Model comparison and model selection

After the review work and finding the models that were most repeated, the performance of those machine learning models was compared for the task of fake job detection. The models considered for comparison were logistic regression, random forest, support vector machines (SVM), and the passive aggressive classifier. The evaluation was performed using a test size of 0.2, meaning that 20% of the dataset was reserved for testing purposes while the remaining 80% was used for training. The Tf-idf was used for feature extraction and the total features were 64867.

The results of the model comparison revealed the following accuracies for each algorithm: logistic regression achieved an accuracy of 95.48%, random forest achieved an accuracy of 98.1%, SVM achieved an accuracy of 97.90%, and the passive aggressive classifier achieved the highest accuracy of 98.49%.

The logistic regression model, with an accuracy of 95.48%, performed relatively well in classifying fake job postings. However, the random forest model outperformed it with an accuracy of 98.1%, indicating its ability to capture more complex relationships within the data. SVM also showed strong performance with an accuracy of 97.90%, demonstrating its effectiveness in separating the two classes. However, it was the passive aggressive classifier that achieved the highest accuracy of 98.49%, suggesting its superior capability in accurately identifying fake job postings.

These results indicate that the passive aggressive classifier is the most suitable model for fake job detection in this project, as it achieved the highest accuracy among the evaluated models. It exhibits a robust ability to classify job postings as genuine or fake, thereby demonstrating its potential for effective deployment in real-world scenarios.

4.3 Results Snapshot

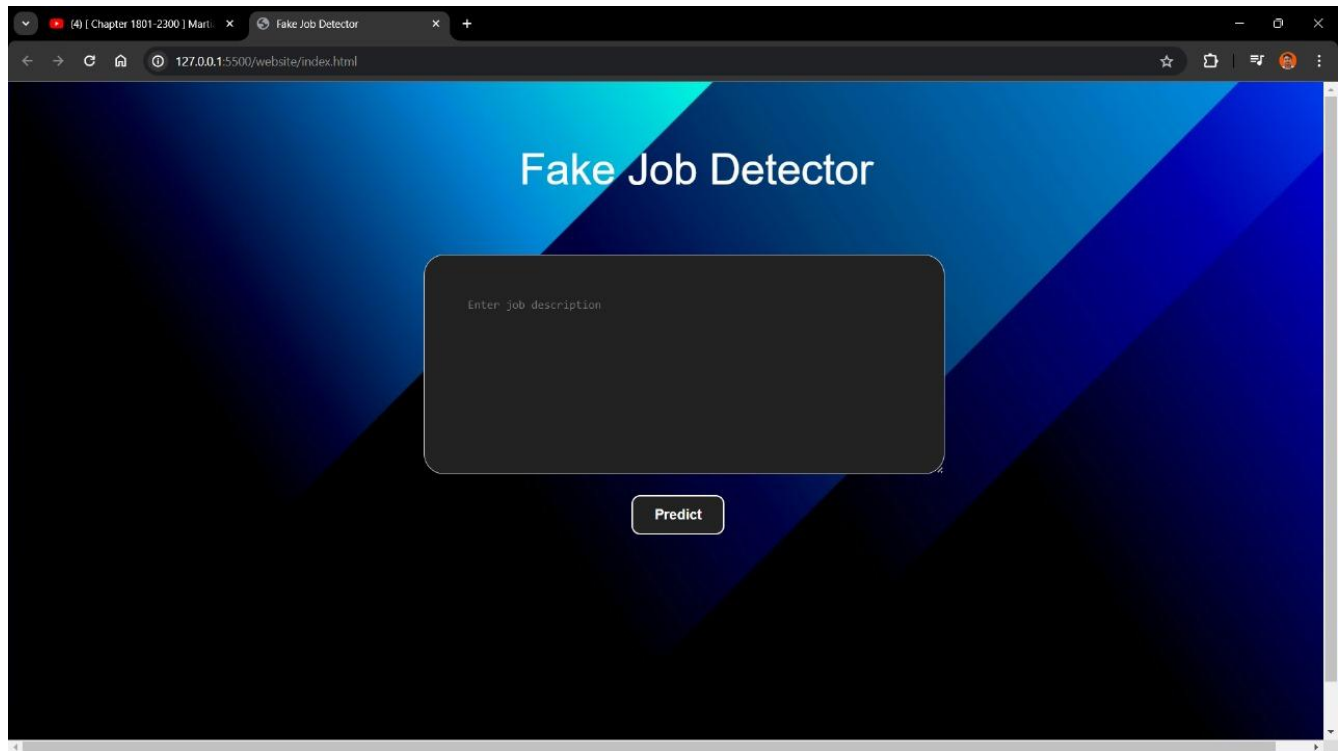


Figure 29: Basic Interface

- ➔ Here we can see a basic interface of the front-end of our project.
- ➔ Here we have a Title on the top of the web page
- ➔ Here we can see a space where the user can pass their input for the job description in order to check
- ➔ Beneath it we can see a button stating “Predict” when clicked it can predict the job if fraudulent or not.

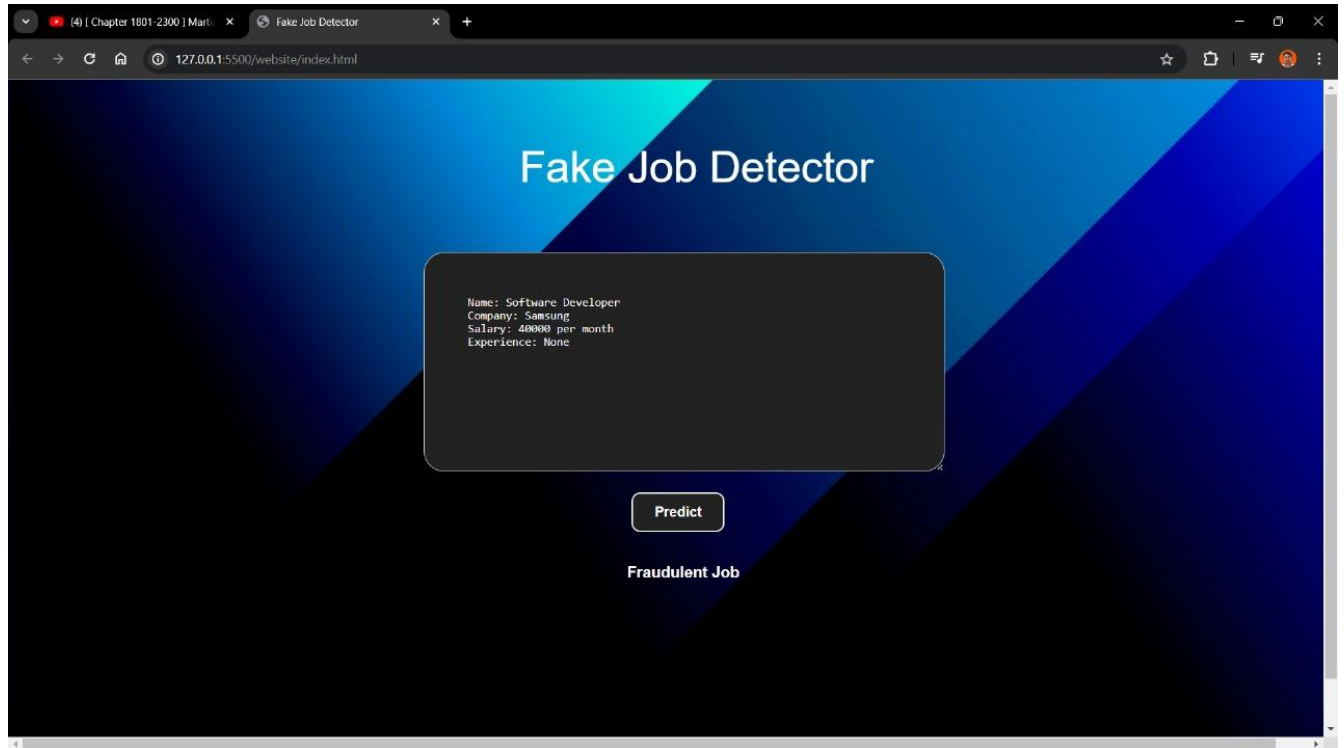


Figure 30: Fraudulent Job

➔ Here we can see a fraudulent job that we took as an example and we can observe that our model predicted it perfectly as a fraudulent job.

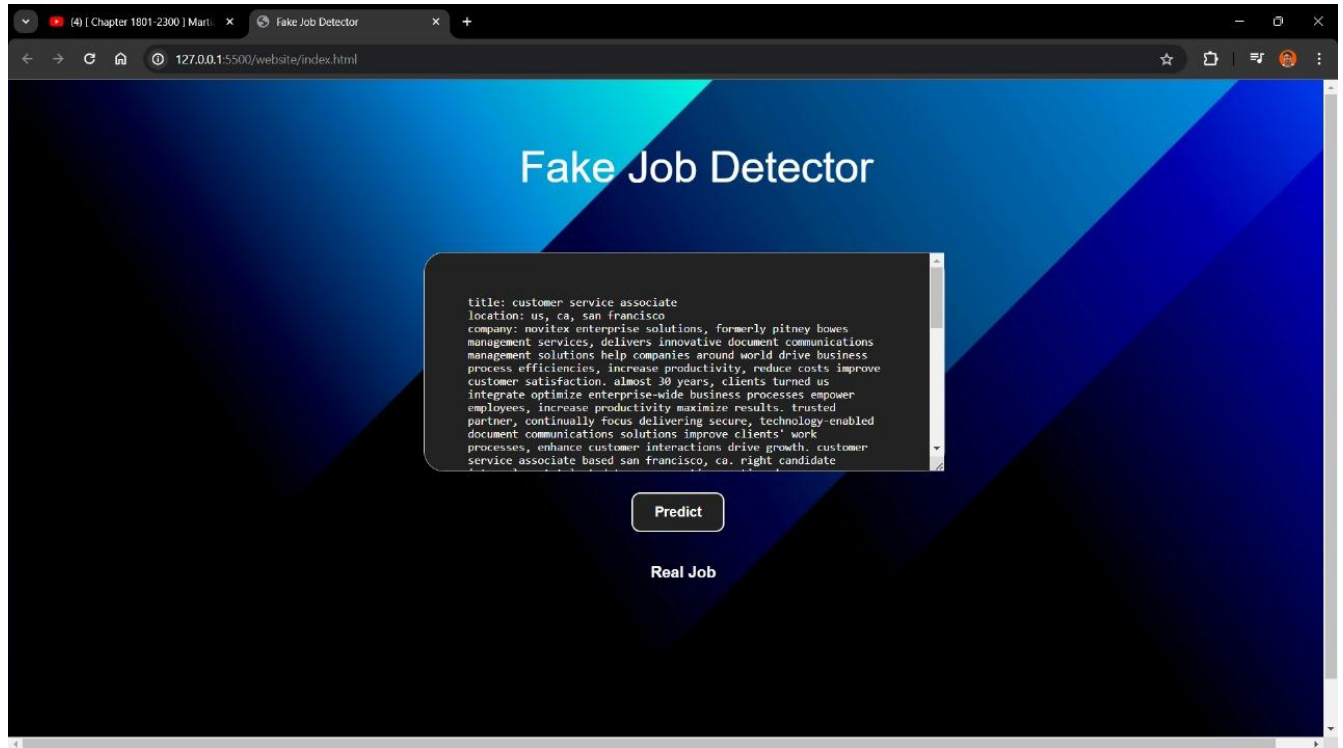
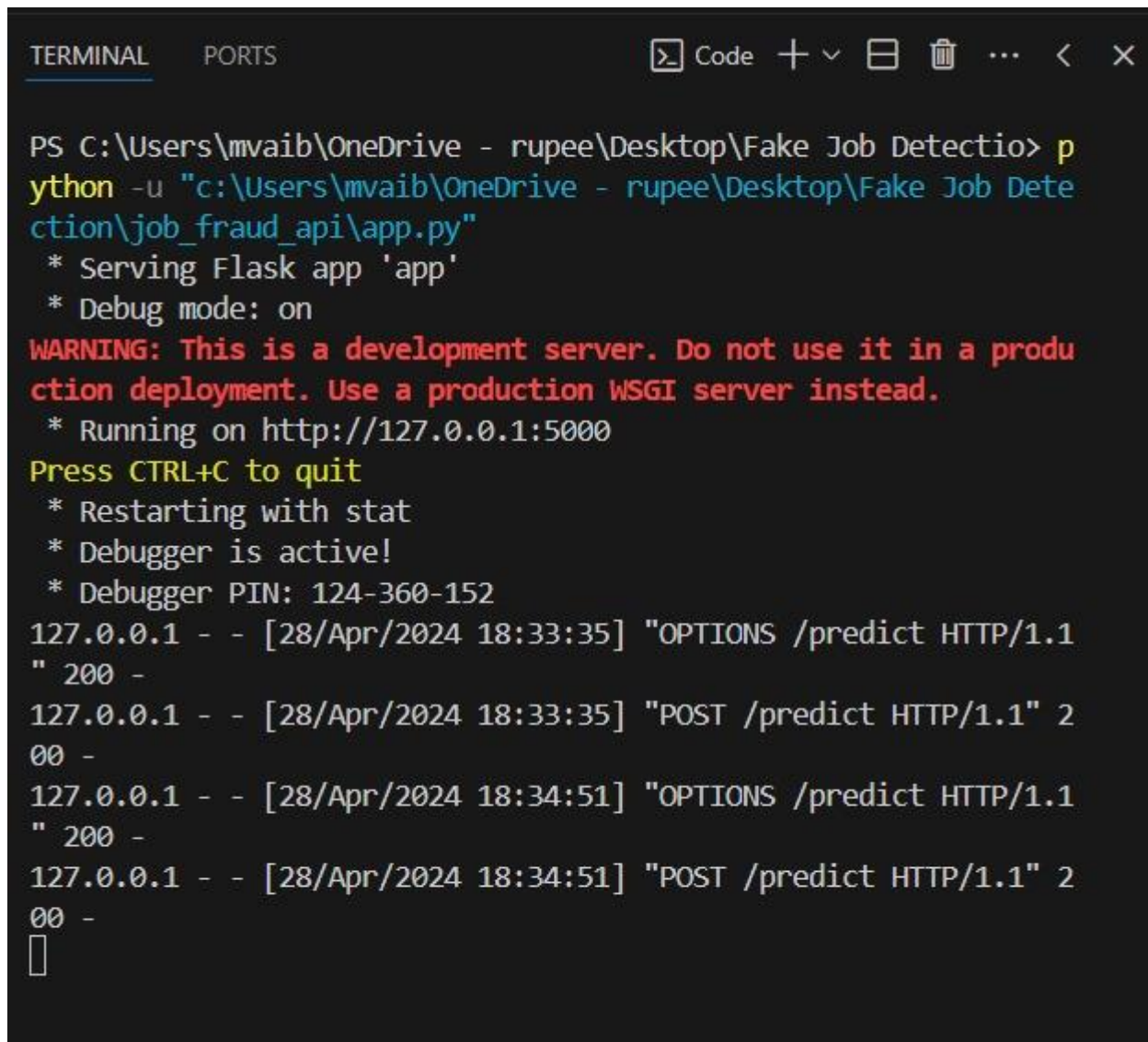


Figure 31: Real Job

- ➔ Here we can see a real job that we took as an example and we can observe that our model predicted it perfectly as a real job.



The image shows a terminal window within an IDE. The window has a dark background and a light-colored border. At the top, there are tabs labeled 'TERMINAL' and 'PORTS'. To the right of the tabs are several icons: a code icon, a plus sign, a checkmark, a square, a trash can, an ellipsis, a left arrow, and a close button. The terminal content shows a command prompt where a Python script is being run. The output includes status messages, a warning about the development server, and several HTTP request logs.

```
PS C:\Users\mvaib\OneDrive - rupee\Desktop\Fake Job Detectio> p
ython -u "c:\Users\mvaib\OneDrive - rupee\Desktop\Fake Job Dete
ction\job_fraud_api\app.py"
* Serving Flask app 'app'
* Debug mode: on
WARNING: This is a development server. Do not use it in a produ
ction deployment. Use a production WSGI server instead.
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
* Restarting with stat
* Debugger is active!
* Debugger PIN: 124-360-152
127.0.0.1 - - [28/Apr/2024 18:33:35] "OPTIONS /predict HTTP/1.1
" 200 -
127.0.0.1 - - [28/Apr/2024 18:33:35] "POST /predict HTTP/1.1" 2
00 -
127.0.0.1 - - [28/Apr/2024 18:34:51] "OPTIONS /predict HTTP/1.1
" 200 -
127.0.0.1 - - [28/Apr/2024 18:34:51] "POST /predict HTTP/1.1" 2
00 -
█
```

Figure 32: Terminal

➔ Here we can see our terminal in the IDE

CHAPTER 5

5 CONCLUSION AND FUTURE WORK

In conclusion, the primary objective of this project was to develop a fake job detection system using machine learning methodologies. Through an exhaustive review of seven research papers, we gained invaluable insights into prevailing approaches and algorithms within the realm of fake job detection. By curating a diverse dataset of job postings and subjecting it to preprocessing, we established a robust system architecture. Leveraging feature extraction techniques like TF-IDF and implementing the Passive Aggressive Classifier as the machine learning model yielded promising outcomes, accurately discerning between genuine and fake job advertisements.

The experimental analysis and subsequent evaluation of the system underscored its efficacy in detecting fake job postings. Nevertheless, avenues for future enhancement and exploration remain. Exploring advanced natural language processing (NLP) techniques, such as word embeddings or deep learning models, holds promise for enhancing the system's understanding and analysis of job descriptions. Moreover, addressing the challenge of imbalanced data through techniques like oversampling, undersampling, or synthetic data generation warrants further investigation. Additionally, leveraging ensemble learning methods and integrating real-time data stream analysis capabilities could augment the system's accuracy and resilience. The integration of user feedback mechanisms and continuous model refinement based on evolving patterns and trends in fake job postings also holds potential for further improving system performance.

In summary, we successfully reviewed seven pertinent papers and presented our findings at MACORE 2023, securing paper acceptance for IJERA. Additionally, we developed and evaluated a fake job detection system. The future trajectory of this endeavor lies in exploring advanced NLP techniques, addressing data imbalance, leveraging ensemble learning, integrating real-time data analysis, incorporating user feedback mechanisms, and considering deployment and scalability aspects. Through continued exploration of these avenues, the fake job detection system can evolve to further mitigate fraudulent job advertisements in the digital job market.

In conclusion, the core aim of this project was to devise a fake job detection system leveraging machine learning methodologies. Through an exhaustive review of seven research papers, we gained deep insights into the existing approaches and algorithms in fake job detection. By assembling a diverse dataset of job postings and preprocessing it meticulously, we established a robust system architecture. Leveraging feature extraction techniques like TF-IDF and adopting the Passive Aggressive Classifier as the machine learning model yielded promising results in accurately distinguishing between genuine and fake job advertisements.

The experimental analysis and subsequent evaluation of the system showcased its efficacy in identifying fake job postings. However, there exist several avenues for future improvement and exploration. The exploration of advanced natural language processing (NLP) techniques, such as word embeddings or deep learning models, holds promise for enhancing the system's understanding of job descriptions. Furthermore, addressing data imbalance through techniques like oversampling, undersampling, or synthetic data generation warrants further investigation. Additionally, the application of ensemble learning methods and the integration of real-time data stream analysis capabilities could potentially enhance the system's accuracy and robustness. Incorporating user feedback mechanisms and continuously refining the model based on emerging patterns and trends in fake job postings are essential for further improving system performance.

In summary, we successfully reviewed seven relevant papers and presented our findings at MACORE 2023, securing paper acceptance for IJERA. Furthermore, we developed a fake job detection system and evaluated its performance. The future trajectory of this project involves exploring advanced NLP techniques, addressing data imbalance, leveraging ensemble learning, integrating real-time data analysis, incorporating user feedback mechanisms, and considering deployment and scalability aspects. Through continued exploration of these avenues, the fake job detection system can evolve to better combat fraudulent job advertisements in the digital job market.

TENTATIVE CHAPTER PLAN FOR THE PROPOSED WORK

CHAPTER1: INTRODUCTION

Here we are providing with the basic overview of the project

CHAPTER2: LITERATURE REVIEW

Here we are analyzing the existing literature solutions

CHAPTER3: DESIGN FLOW/PROCESS

Here we are discussing about the blueprint of the project and understand its working

CHAPTER4: RESULT ANALYSIS AND VALIDATION

Here we will take a look the results of our project and required validations

CHAPTER5: CONCLUSION AND FUTURE WORK

It defines the outcome and the future scope of our project

TENTATIVE CHAPTER PLAN FOR THE PROPOSED WORK

Contains the overview of the chapters in the report

REFERENCES

Contains the references used in the report

REFERENCES

- Baraneetharan, None E. “[PDF] Detection of Fake Job Advertisements Using Machine Learning Algorithms by None E. Baraneetharan · 10.36548/Jaicn.2022.3.006 · OA.Mg.” OA.Mg · Open Access for Everyone · Download and Read over 240 Million Research Papers, 2022, <https://oa.mg/work/10.36548/jaicn.2022.3.006>.
- Ullah1, Zahid. “A Smart Secured Framework for Detecting and Averting Online Recruitment Fraud Using Ensemble Machine Learning Techniques [PeerJ].” PeerJ Computer Science, 8 Feb. 2023, <https://peerj.com/articles/cs-1234/>.
- Naudé, M., Adebayo, K.J. and Nanda, R. (2022) A machine learning approach to detecting fraudulent job types - ai & society, SpringerLink. Springer London. Available at: <https://link.springer.com/article/10.1007/s00146-022-01469-0>.
- Vo, A. and Sharma, R. (2021) Dealing with the class imbalance problem in the detection of fake job ... Available at: https://www.researchgate.net/publication/350399453_Dealing_with_the_Class_Imbalance_Problem_in_the_Detection_of_Fake_Job_Descriptions.
- Amaar, A. et al. (2022) Detection of fake job postings by utilizing machine learning and ... Available at: https://www.researchgate.net/publication/357576631_Detection_of_Fake_Job_Postings_by_Utilizing_Machine_Learning_and_Natural_Language_Processing_Approaches.
- Kumar, D.V. (2020) Classifying fake and real job advertisements using machine learning, Analytics India Magazine. Available at: <https://analyticsindiamag.com/classifying-fake-and-real-job-advertisements-using-machine-learning/>.

- Shawni Dutta, Prof.Samir Kumar Bandyopadhyay "Fake Job Recruitment Detection Using Machine Learning Approach" International Journal of Engineering Trends and Technology 68.4(2020):48-53.